Process Modeling of Deposit Solidification in Droplet Based Manufacturing

by

Paul J. Acquaviva

B.S., Mechanical Engineering
Clarkson University, May, 1989

Submitted to the Department of Mechanical Engineering
in Partial Fulfillment of the Requirements for the Degree of
MASTERS DEGREE OF SCIENCE IN MECHANICAL ENGINEERING

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1995

Signature of Author: _____
Department of Mechanical Engineering
June 1995

Certified by: _____
Professor Jung-Hoon Chun
Thesis Supervisor

Accepted by: _____
Professor Ain A. Sonin, Chairman
Graduate Committee, Department of Mechanical Engineering

Process Modeling of Deposit Solidification in Droplet Based Manufacturing

by
Paul J. Acquaviva

Submitted to the Department of Mechanical Engineering
on May 8, 1995 in partial fulfillment of the
requirements for the Degree of Master of Science

ABSTRACT

The uniform droplet spray (UDS) forming process has been developed to enable precise control of droplet thermal states and the resultant material microstructure of the deposit. By having a uniform droplet size throughout the spray, all the droplets deposited onto a substrate will have the same thermal state upon impact, allowing for precise control of the solidification process within both the droplets and the deposit.

In this study, a one-dimensional, finite difference model was created to predict the temperature and liquid fraction of the deposit during the UDS process. The model employs an explicit temperature-enthalpy method to incorporate a variety of solidification assumptions. Experiments were conducted using Sn-15wt.%Pb binary alloy to validate the model. Temperatures were measured in the deposit and acceptable agreement with the simulation was obtained.

Modeling has shown that the deposit thermal state is highly dependent on variations in spray conditions, which are predicted using droplet trajectory and droplet thermal models. Further modeling of the individual droplet splats immediately after impact revealed that three phases of solidification exist: droplet solidification; droplet splat solidification; and consolidated deposit solidification, each with significantly different cooling rates. By manipulating process parameters, the percentage of solidification within each of these three stages can be shifted, resulting in changes in final microstructure. By using the deposit thermal model, the relationship between process parameters and solidification behavior can be understood to obtain the desired microstructure and material properties.

Thesis Advisor: Dr. Jung-Hoon Chun
Title: Edgerton Associate Professor, Department of Mechanical Engineering

Acknowledgments

I would first and foremost like to thank my wife, Susan for her enduring patience and moral support over the past two years. Secondly, a big thank you to my family who has worked around my schedule for years now. I would like to give special thanks to my advisor, Professor Chun for providing me with this great learning experience. A sincere thank you to General Electric for its financial support with special thanks to Dennis Evans, Mike Hill, Joe Bussichella, Sam Spring, Andy Olsen, and Joan Lewis for their efforts in helping me finish this ordeal. To my lab partners who I have learned so much from: to Chen-An and Juneau Chen for enduring the all-night experiment sessions and sometimes painful debates; to Ho-Young Kim for his help in preparing this manuscript; thanks to Suk-Young Chey, Jeanie Cherng, and Randy Hyun for their patience in sharing the lab facilities; to Pyongwon Yim, Godard Abel, and Wesley Williams for their part in building the UDS apparatus used in my experiments; and thanks to Tom Nowak for all his words of wisdom, may he someday break into the number one slot. A word of thanks to the GE probe shop for their lessons on the art of instrumentation. A thank you to Samir Sayegh, for sharing with me his inspirational words. And a sincere thank you to Leslie Regan for her support and patience from start to finish.

Contents

1 INTRODUCTION 8
1.1 Motivation 8
1.2 The UDS process 9
1.3 Research description 11

2 MODEL FORMULATION 14
2.1 Introduction 14
2.2 Governing equations 15
2.3 Solidification model 16
2.4 Boundary conditions 16
2.5 Computational considerations 17

3 EXPERIMENTAL VALIDATION AND MODELING SENSITIVITIES 20
3.1 Experiment description 20
3.2 Experimental results 25
3.3 Simulation results of experiment 26
3.4 Comparison of experiment to simulation 28
3.5 Sensitivities analysis 30

4 MICROMODELING OF DROPLET SPLATS 36
4.1 Introduction 36
4.2 Assumptions 36
4.3 Computational considerations 38
4.4 Simulation results 38
4.5 Discussion 40

5 DEPOSIT MICROSTRUCTURE 43
5.1 Introduction 43
5.2 SEM preparation and sample description 43
5.3 SEM micrograph presentation 44
5.4 Compositional variation 48
5.5 Discussion 48

6 CONCLUDING REMARKS 50
6.1 Conclusions 50
6.2 Recommendations for future research 51

BIBLIOGRAPHY 52

APPENDIX
A Spray forming process comparison 55
B Issues in thermal spray forming 57
C Background of deposition modeling 60
D Model discretization 61
E Operating instructions for model 63
F Simulation input files 65
G Spray model sensitivities 69
H Simulation source code 71

List of Tables

Table 1. Experimental conditions. 23
Table 2. Physical constants of Sn-15 wt.%Pb 26
Table 3. Compositional variations of deposit per SEM analysis 48
Table 4. Thermal spray process breakdown 55
Table 5. Summary of reported as-sprayed deposit porosities 56

List of Figures

Figure 1 The uniform droplet spray apparatus....9
Figure 2 Uniform powder produced using the UDS process....10
Figure 3 Process flow chart....11
Figure 4 Computer models of the UDS process....12
Figure 5 Coordinate system for substrate and deposit....14
Figure 6 Flow chart of deposit solidification model....18
Figure 7 Deposit element schematic....19
Figure 8 Overall experimental setup....20
Figure 9 Substrate setup before experiment began....21
Figure 10 Substrate setup after experiment finished....21
Figure 11 Photograph of the deposit after the deposition experiment....22
Figure 12 Simulated spray radius vs. flight distance....23
Figure 13 Deposit thickness during the experiment....24
Figure 14 Simulated droplet temperature vs. flight distance....24
Figure 15 Measured temperatures in experiment....25
Figure 16 Simulated temperature at thermocouple locations within the deposit....27
Figure 17 Simulated temperature (constant temperature contours)....27
Figure 18 Simulated liquid fraction (constant liquid fraction contours)....28
Figure 19 Comparison of simulations vs. experiment at 28 mm....29
Figure 20 Effect of mass flux changes on the deposit surface temperature....31
Figure 21 Temperature-enthalpy relationships....31
Figure 22 Effect of solidification assumptions on simulation results of the experiment....32
Figure 23 Effect of contact coefficient assumptions on simulation results of the experiment....34
Figure 24 Effect of deposit diameter on solidification on simulation results of the experiment....35
Figure 25 Droplet solidification at a deposit temperature of 153°C....39
Figure 26 Droplet solidification at a deposit temperature of 182°C....39
Figure 27 Droplet surface temperature at different deposit conditions....40
Figure 28 Schematic of deposition microstructure after experiment....44
Figure 29 SEM #1: Positions A and B, 100x magnification....45
Figure 30 SEM #2: Position C, 200x magnification....45
Figure 31 SEM #3: Position C, 600x magnification....46
Figure 32 SEM #4: Position F, 300x magnification....46
Figure 33 SEM #5: Position H, 300x magnification....47
Figure 34 SEM #6: Position J, 300x magnification....47
Figure 35 Subgroups of the thermal spray forming process....56
Figure 36 Ductility of several copper dispersion alloys vs. volume fraction....58
Figure 37 Comparison of reported range in droplet size for various spraying processes....59
Figure 38 Spray sensitivity on charging....70
Figure 39 Spray sensitivity on instability perturbation....70

Nomenclature

A_s	top surface area of the deposit, m^2
c_p	specific heat, $J / kg\ °C$
C_o	initial concentration of solute.
C_ℓ	concentration of the solute in the liquid phase
C_s	concentration of the solute in the solid phase
d_d	droplet diameter, m
f_ℓ	liquid fraction
f_p	droplet generation frequency, sec^{-1}
h	convection heat transfer coefficient at the surface, $W / m^2\ °C$
h_c	contact coefficient at the deposit interface, $W / m^2\ °C$
h_{eq}	equivalent contact coefficient of a droplet, $W / m^2\ °C$
H	enthalpy, J / kg
H_d	enthalpy of the droplets upon impact, J / kg
H_ℓ	enthalpy of the liquid, J / kg
H_s	enthalpy of the solid, J / kg
H_{sur}	enthalpy of the deposit at the surface, J / kg
k	thermal conductivity, $W / m\ °C$
k_ℓ	liquid thermal conductivity, $W / m\ °C$
k_s	solid thermal conductivity, $W / m\ °C$
K	partition coefficient
$\dot{m}$	average mass flow rate of the metal spray, kg / sec
t	time, sec
T	temperature, $°C$
T_o	temperature at the bottom of the substrate, $°C$
T_g	temperature of the inert gas, $°C$
T_ℓ	liquidus temperature of the alloy with the initial concentration, C_o , $°C$
T_r	reference temperature, $°C$
T_s	deposit surface temperature, $°C$
T_M	melting temperatures of the primary phase, $°C$
$\Delta H_{f,1}$	latent heat of fusion for the primary species, J / kg
$\Delta H_{f,2}$	latent heat of fusion for the secondary species, J / kg
Δt	time step, sec
Δz	increase in element size, m
ε	emissivity of the deposit surface
ξ	deposit thickness at time t, m
ξ_s	droplet splat thickness, m
ζ	substrate thickness, m
ρ	density, kg / m^3
σ	Stefan-Boltzmann constant, $W / m^2 K^4$

Chapter 1

INTRODUCTION

1.1 Motivation

For the production of parts or materials with optimal properties, spray forming offers an attractive alternative to conventional casting. This is due in large part to rapid solidification within the droplets during flight and incremental solidification at the deposit surface [1, 2]. Through incremental solidification, the solidification rate is directly controlled, resulting in control of the material microstructure. Numerous studies have shown that these processes can produce rapidly solidified materials with uniform, fine, equiaxed microstructure and little phase segregation in a variety of alloy systems [3-10]. Although these studies are promising, substantial porosity and material property variation are present in the deposit. These problems are a result of two separate issues:

1) Process complexity which makes control difficult, and
2) A lack of applied process modeling for understanding of the physics within the process.

For these reasons, the full potential of spray forming concept has not been realized [11].

The problem with most metal spray processes is the inherently coupled nature of the delivery system (see Appendix A and B). Manipulating any one process parameter affects several other parameters, so that certain ideal conditions are difficult to obtain. Spray atomization is an example of this. This process relies upon an impinging gas jet to break up the molten stream and cool the droplets. If the parameter which needs changing is droplet size, this may require a higher velocity jet. This change would also cause the drops to cool quicker and disperse wider. Therefore, the process may not have the capability to independently adjust parameters needed to achieve the optimal condition. It is for this reason that, in practice, processes are optimized by adjusting independent process-specific parameters such as spray distance or the ratio of gas to metal spray. Ideally, however, the statistical study should be based on physical parameters within the system, such as droplet temperature. Physical parameters have a direct influence on the material microstructure. Knowledge-based optimization will result from understanding them.

Process modeling of existing spray processes is difficult due to the complexity and variation within the spray. Gas atomization, for example, results in a spray with a wide distribution of droplet diameters and trajectories. Droplets of different sizes will solidify at

different rates, resulting in a spray which contains liquid and partially liquid droplets and solid particles. The relative gas velocity and turbulence intensity within the spray also varies, adding further to the droplet thermal state variation. Therefore, only an averaged droplet thermal state can be used in characterizing the process [12, 13]. Due to the lack of a rigorous method to measure the droplet temperature in situ, these calculations cannot be validated and therefore should be questioned.

Because of these complexities, the thermal spray industry remains heavily reliant upon recipes and process-specific parameters determined through experimental trial and error. In many cases, this practice is the only practical option, resulting in a lack of understanding of the physical phenomenon that determines the final product. These problems have resulted in a restricted range of attainable microstructures, frequent problems of porosity [10, 11], and limited utilization of process models.

The UDS process was developed to overcome these shortcomings [14-15]. The uniform droplet size within the spray provides a consistent and uniform droplet thermal state upon impact, allowing for simplified modeling, process control, and characterization [16].

1.2 The UDS process

The UDS process employs the concept of laminar jet instability to create a uniform droplet metal spray. A schematic diagram of the UDS apparatus is shown in Fig. 1.

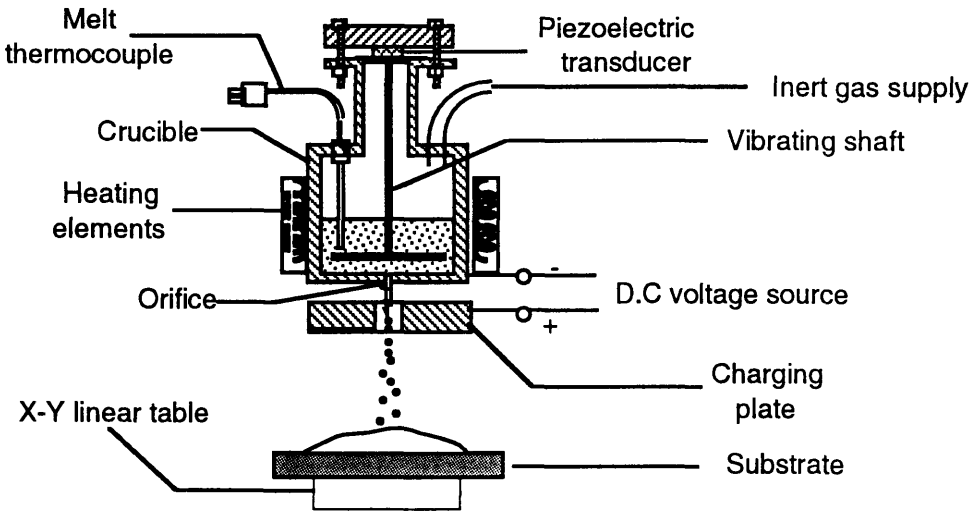


Fig. 1. The uniform droplet spray apparatus

The molten metal contained in a heated crucible is ejected with applied pressure through a small orifice at the bottom of the crucible, forming a laminar jet. The jet is broken at the

desired frequency by vibrations generated by a piezoelectric transducer to form a stream of uniform droplets. As it breaks from the jet, each droplet is then electrically charged by a DC voltage charging plate to prevent merging with other droplets. Having the same polarity, the droplets repel each other and form a diverging spray. The droplets are then deposited onto a temperature- and motion-controlled substrate to produce the desired deposit shape and microstructure. An example of the solidified droplets produced with the UDS process is given in Fig. 2.

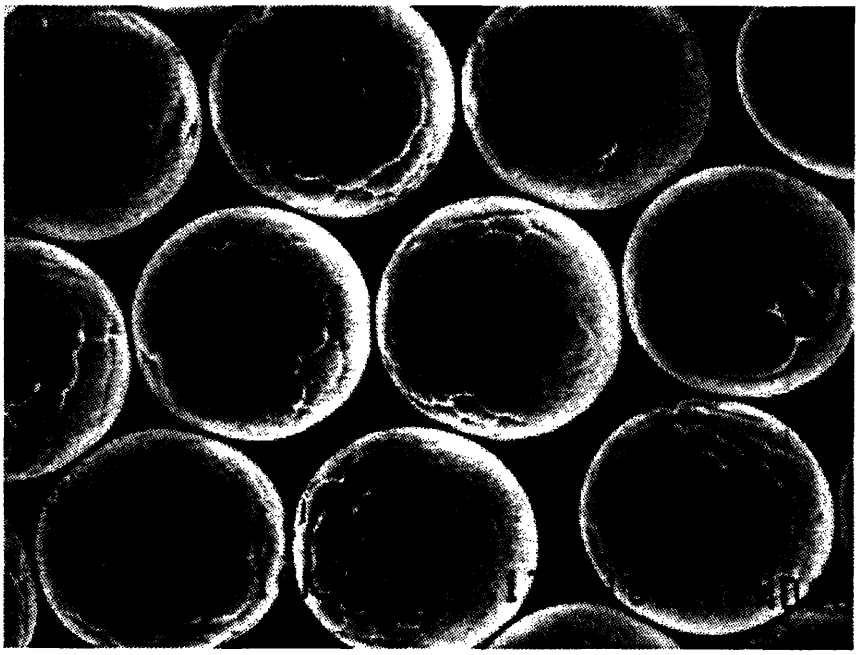


Fig. 2. Uniform powder produced using Uniform Droplet Spray.

The geometry and microstructure are controlled by three key design parameters: 1) the mass flux distribution to the substrate; 2) the droplet thermal state; and 3) the deposit thermal state. The relationships between the controllable process parameters and these design parameters are illustrated in Fig. 3. The orifice diameter, jet velocity, and break-up frequency determine the droplet diameter, while the crucible pressure controls the initial velocity of the droplets. The combination of initial velocity, charging voltage, and droplet diameter determines the spray cone angle and the mass flux distribution. Substrate distance and motion can then be manipulated to control the geometry of the deposit. The melt temperature and droplet velocity determine the temperature of the

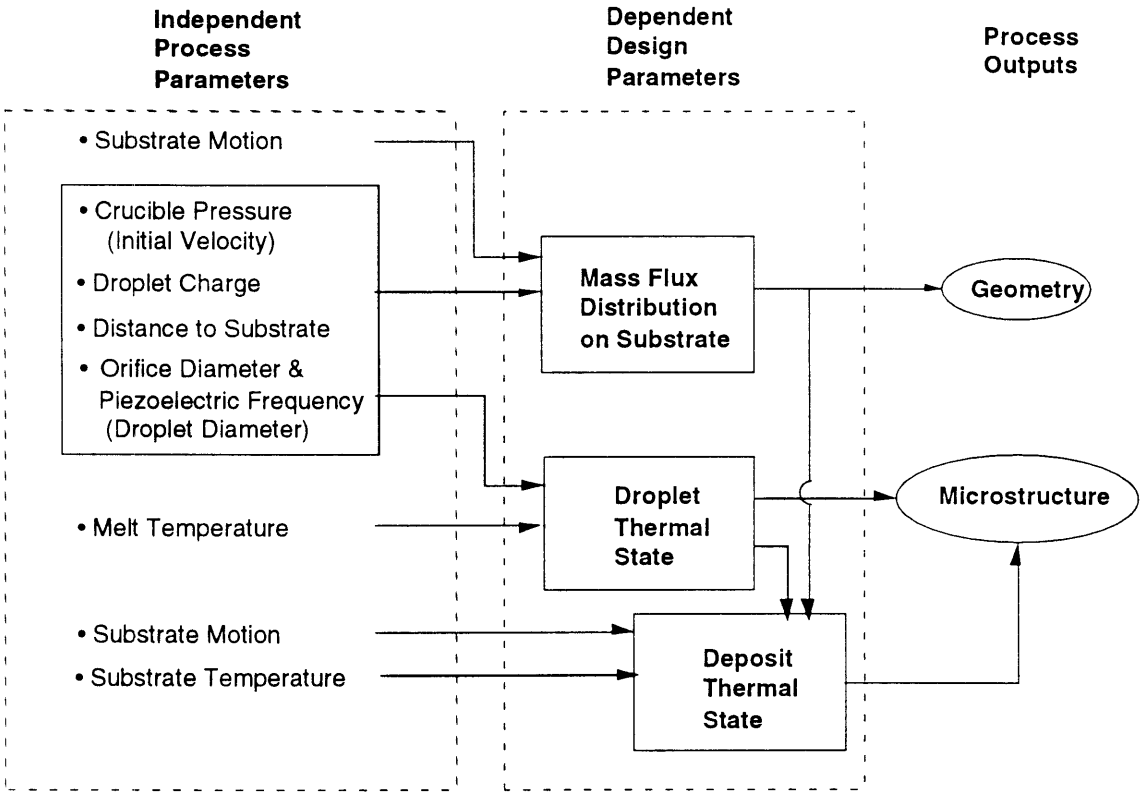


Fig. 3. Process flow chart

droplets along their flight path. With the droplet enthalpy and mass flux determined, the substrate temperature can be manipulated to control the thermal state of the deposit.

1.3 Research description

The research involves modeling and experimentation of the UDS process. Specifically, a computer model has been generated to simulate the heat transfer and solidification within the growing deposit. This model serves as a tool for predicting and optimizing the thermal conditions at the deposit to obtain the ideal or desired microstructure and the resulting material properties. The formulation of this model is described in chapter 2.

Experiments were performed using Sn-15wt.%Pb alloy to validate the deposit thermal model. In the validation process, other models were utilized to simulate the jet breakup [17], droplet trajectory [18], and droplet thermal state [19] within the spray. Fig. 4 illustrates schematically the relative order and scope of the three models. These models were modified to obtain two critical spray parameters at the point of impact: the droplet thermal state and the spray geometry.

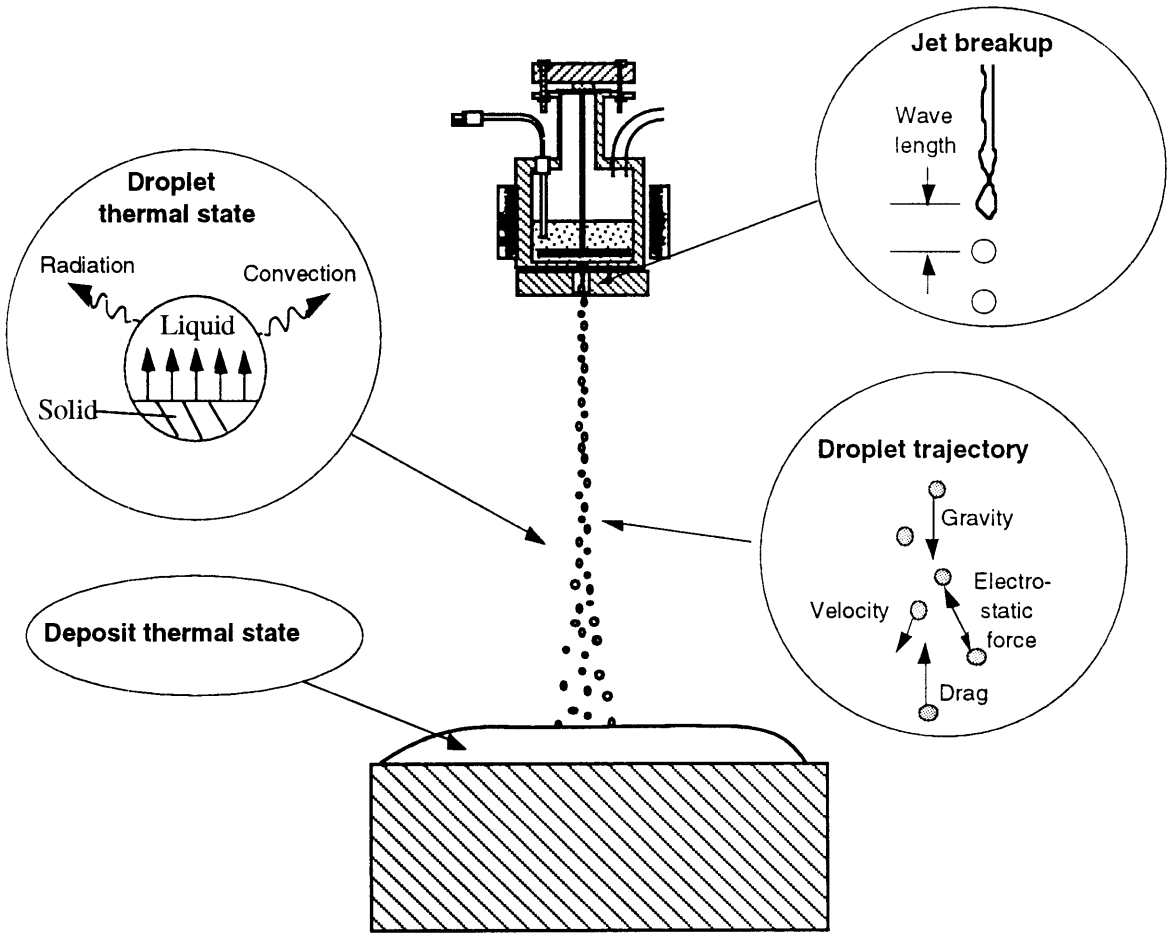


Fig. 4. Computational models within the UDS process

The impact of changes to the spray conditions on the deposit thermal state have been explored. In comparing the experiments with the simulation, the impact of several other assumptions are quantified. The experimental validation and sensitivities are given in chapter 3.

The droplet model was also utilized to model the rapid solidification which occurs within the individual droplet splats directly after impact. The difference between this modeling and that of the overall deposit is in the time and length scale of the simulation. The validation of the experiment considered the macroscopic or "time averaged" deposit thermal state. This modeling treats the spray as continuous with perfectly distributed mass flow within the spray. In reality, the spray is comprised of discrete droplets which impact, cool, and solidify independently until they reach the deposit surface temperature. The microscopic model predicts the rate of solidification of the individual droplet splat,

revealing the extent of solidification which occurred rapidly. This analysis is described in chapter 4.

Chapter 5 examines the microstructure of the deposit created during the validation experiment. The microstructure includes both the grain morphology and the compositional variation within both phases. SEM (Scanning Electron Microscope) analysis is performed to demonstrate the variation of material microstructure due to the variation of deposit conditions which occurred during the experiment. The composition of the deposit has been examined to gain insight into the solidification assumptions which influence the results of the deposit thermal model.

Chapter 2

MODEL FORMULATION

2.1 Introduction

During spray deposition, droplets land on the deposit surface one by one, generating discrete mass and enthalpy fluxes. However, if the deposition rate is high and only the time-averaged thermal state in the deposit is desired, the thermal modeling can be simplified by treating the discrete mass and enthalpy flux as continuous. That is the assumption in this model as well as in other existing models of spray forming deposition (see Appendix C). Other assumptions made to simplify the model are that: (1) convective liquid flow within the deposit is negligible, i.e., only conduction is considered; and (2) transverse thermal gradients within the deposit and substrate are negligible. Therefore, a one-dimensional model is used. The coordinate system employed for modeling deposit solidification and growth is shown in Fig. 5.

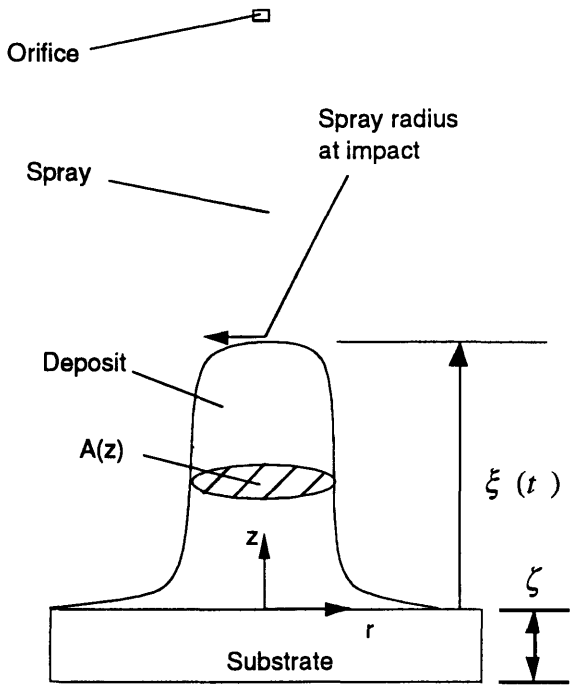


Fig. 5. Coordinate system for substrate and deposit

2.2 Governing equations

The deposit thermal model can be formulated by considering the balance between enthalpy change and conduction heat transfer at any location within the deposit and the substrate. The governing energy equation can be described as:

$$\frac{\partial}{\partial z}\left(k\ \frac{\partial T}{\partial z}\right) = \rho\ \frac{\partial H}{\partial t} \tag{1}$$

where k is the thermal conductivity, T is the temperature, ρ is the density, and H is the enthalpy.

For the substrate, the thermal conductivity and specific heat are assumed to be independent of temperature, and the enthalpy is a function of the temperature only; therefore, the enthalpy can be written as:

$$H = c_p\ (T - T_r) \qquad \text{for } z < 0 \tag{2}$$

where c_p is the specific heat of the substrate and T_r is the reference temperature.

For the deposit, the thermal conductivity and the enthalpy are functions of the liquid fraction, f_ℓ, and are assumed to be:

$$k\ (f_\ell) = f_\ell\, k_\ell + (1-f_\ell)\ k_s\ , \qquad \text{for } z > 0 \tag{3}$$

$$H = f_\ell H_\ell + (1 - f_\ell) H_s \qquad \text{for } z > 0 \tag{4}$$

where k_ℓ and k_s, respectively, are the liquid and solid thermal conductivities, and H_ℓ and H_s are the enthalpies of the liquid and solid, respectively. While the enthalpy of the solid deposit is only a function of the specific heat, that of the liquid deposit must also account for the latent heat of fusion. For a binary alloy, they are respectively assumed to be:

$$H_s = \int_{T_r}^{T} c_p(C_s, T\ )\, dT\ , \tag{5}$$

$$H_\ell = \int_{T_r}^{T} c_p(C_\ell, T\ )\, dT + (1-C_\ell)\, \Delta H_{f,1} + C_\ell\, \Delta H_{f,2} \tag{6}$$

where C_s is the concentration of the solute in the solid phase, C_ℓ is the concentration of the solute in the liquid phase, and $\Delta H_{f,1}$ and $\Delta H_{f,2}$ are the latent heats of fusion of the primary and secondary species, respectively. The enthalpy change resulting from the heat of mixing of the two elements is considered later in Chapter 4.

Knowledge of the temperature-enthalpy relationship is required to solve equation (1). This relationship is determined by the solidification model employed.

2.3 Solidification model

The Scheil equation, which assumes no diffusion in the solid phase, complete diffusional mixing in the liquid phase, and local equilibrium at the solid-liquid interface, is adopted here for binary alloy solidification. In the Scheil equation, the interface compositions can be written in terms of liquid fraction [20] as:

$$C_\ell = C_o f_\ell^{(K-1)}, \tag{7}$$

$$C_s = K\, C_o f_\ell^{(K-1)} \tag{8}$$

where C_o is the overall concentration of solute and K is the equilibrium partition coefficient. By assuming the solidus and liquidus slopes are constant, the liquid fraction can be expressed in terms of temperature [20] as:

$$f_\ell = \left(\frac{T_M - T}{T_M - T_\ell}\right)^{-1/(1-K)} \tag{9}$$

where T_M and T_ℓ are the melting temperatures of the primary phase and liquidus temperature of the alloy, respectively. By combining equations (4) through (9), the explicit temperature-enthalpy relationship can be obtained.

The solidification model presented here can be easily adjusted through the temperature-enthalpy relationship. Other solidification models can be employed which may account for dependencies in thermo-physical properties, the impact of the heat of mixing, or non-equilibrium solidification.

2.4 Boundary conditions

At the top surface of the deposit, the boundary condition is determined by the balance of energy across the surface (which includes conduction into the deposit), the enthalpy flux from the spray, and convection and radiation to the surroundings:

$$\left(k\ (T) \frac{\partial T}{\partial z} \right)_{z=\xi} = \frac{\dot{m}\,(H_d - H_s)}{A_s} - h\ (T_s - T_g) - \sigma \varepsilon \left(T_s^4 - T_g^4 \right) \qquad (10)$$

where ξ is the deposit thickness at time, t, $\dot{m}$ is the average mass flux of the metal spray, H_d is the enthalpy of the droplets at impact, H_s is the enthalpy of the deposit at the surface, A_s is the top surface area of the deposit, h is the convection heat transfer coefficient, T_s is the deposit surface temperature, T_g is the temperature of the inert gas, σ is the Stefan-Boltzmann constant, and ε is the emissivity of the deposit surface. Deposition rate and droplet enthalpy, which determine the enthalpy flux, are predicted using the droplet trajectory and thermal models described above. Since the control temperature at the bottom of the substrate, T_o, is held constant, the boundary condition at the bottom is simply stated as:

$$T = T_o \qquad (11)$$

To determine the deposit thermal state, it is necessary to know the thermal state and spatial distribution of the droplets upon impact. The droplet trajectory model [18] and thermal model [19] have been previously created to simulate the spray and to obtain inputs for the deposit thermal model.

2.5 Computational considerations

The deposit thermal model uses a one-dimensional explicit finite difference method to determine the enthalpy, temperature, and liquid fraction of the deposit. Discretized equations are given in Appendix D. A variable grid is employed to account for the growing deposit. The top element of the deposit grows at each time step by the amount:

$$\Delta z = \frac{\dot{m}}{\rho A_s} \Delta t \qquad (12)$$

where Δz is the increase in element size and Δt is the time step. Once the top element reaches a specified size, it is split into two elements, leaving a smaller element at the top, which will continue to grow. The time step is continually adjusted as a function of the smallest element size to minimize computing time by employing the following stability criteria [20]:

$$\Delta t = c\ \alpha \frac{\Delta T}{\Delta z^2} \tag{13}$$

where c is a constant adjusted to 0.003 to provide an accuracy of 99.99%, based on computational trials and α is the thermal diffusivity of the deposit.

A schematic of the model flow chart is shown in Fig. 6. The deposition area is determined by the droplet trajectory model. The rate of deposit thickness growth, $\frac{d\xi}{dt}$, is calculated by considering mass conservation:

$$\frac{d\xi}{dt} = \frac{\pi}{6}\frac{d_d^3}{A_s} f_p \tag{14}$$

where d_d is the droplet diameter and f_p is the droplet generation frequency.

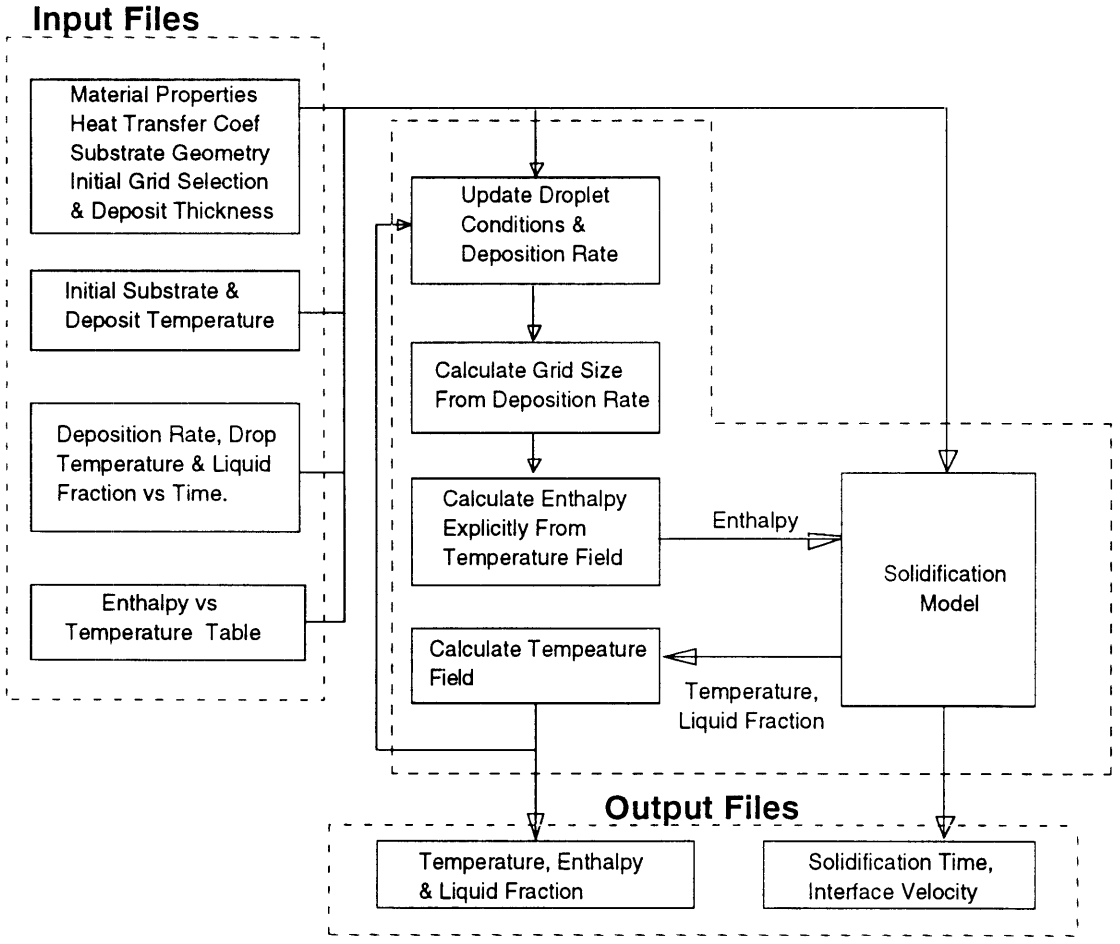


Fig. 6. Flow chart of deposit solidification model

Fig. 7 illustrates the deposit and substrate element grid as well as a schematic of two solidification scenarios. By employing the enthalpy method, the solidification is assumed to be morphology independent, therefore, the solidification front and element grids are independent. If the thermo-kinetics of a planar interface were employed [21], the model would be forced to maintain a distinct interface with no mushy region. This type of model is beneficial in calculating the rate of solidification under significant undercooling, however, it is incorrect when modeling dendritic solidification with a substantial mushy region.

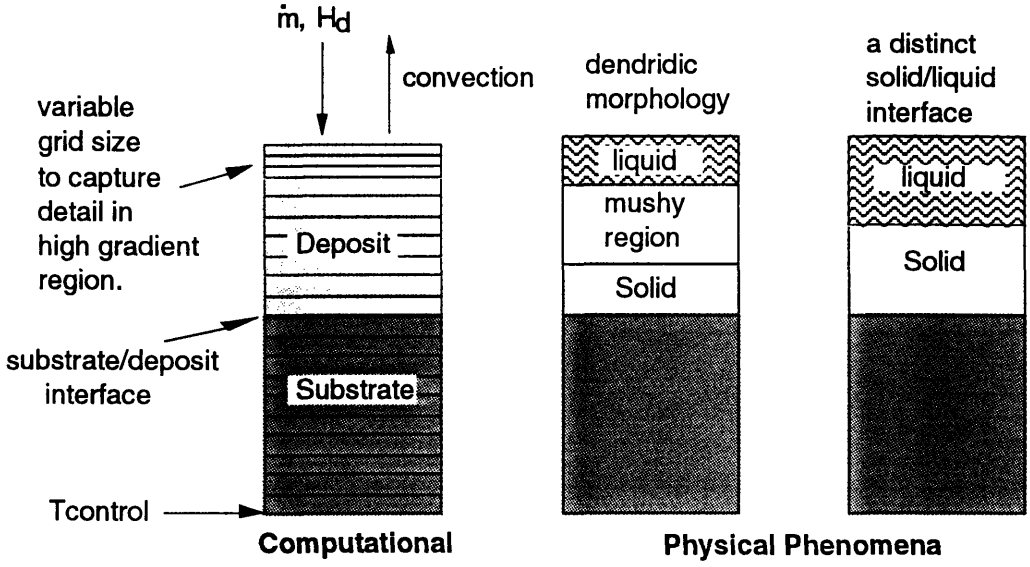


Fig. 7. Deposit element schematic

The model was programmed to be simple to utilize and flexible. Instructions for the model are given in Appendix E. This was accomplished by developing a set of input files, which are listed in Appendix F.

Chapter 3

EXPERIMENTAL VALIDATION

3.1 Experiment description

A schematic of the experimental setup is given in Fig. 8. An instrumented copper substrate, 50 mm in diameter and 25 mm in thickness, was placed on an x-y linear table in a chamber filled with nitrogen gas.

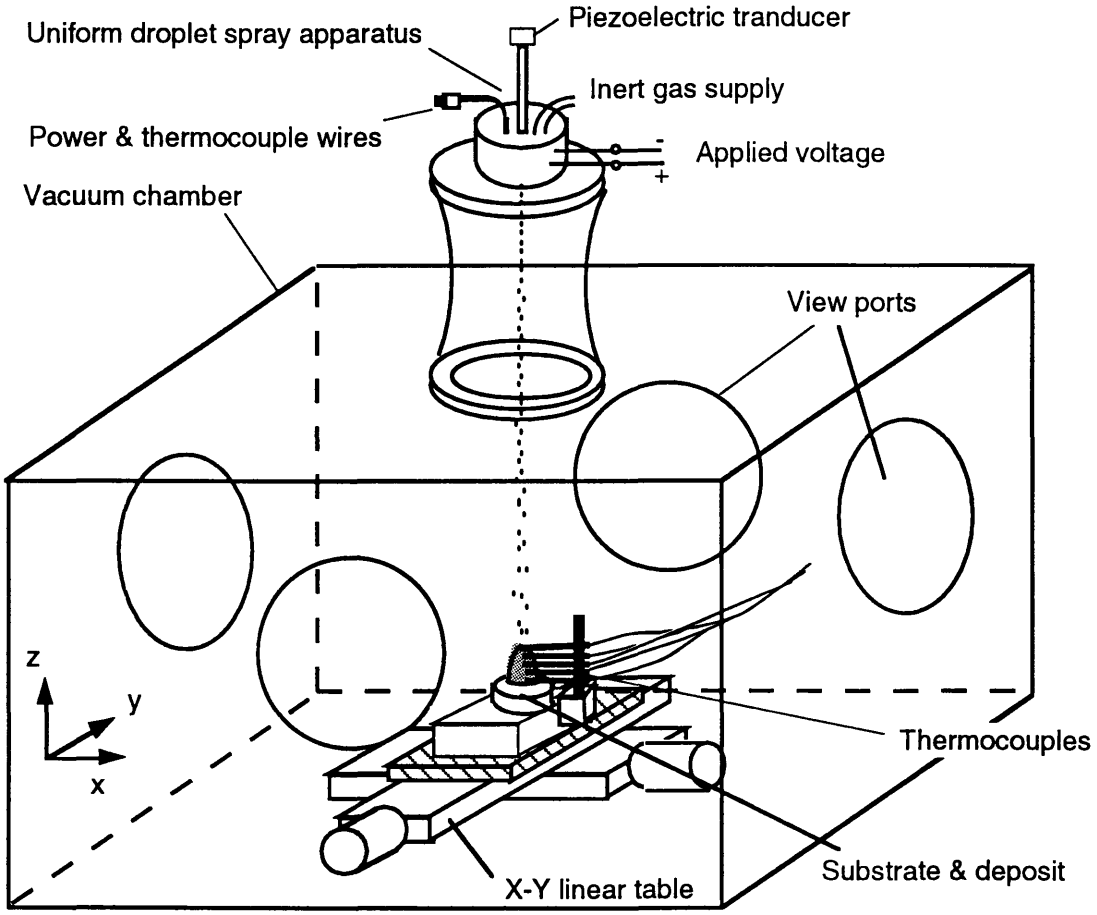


Fig. 8. Overall experimental setup

Figs. 9 and 10 show schematics of the copper substrate before and after the experiment.

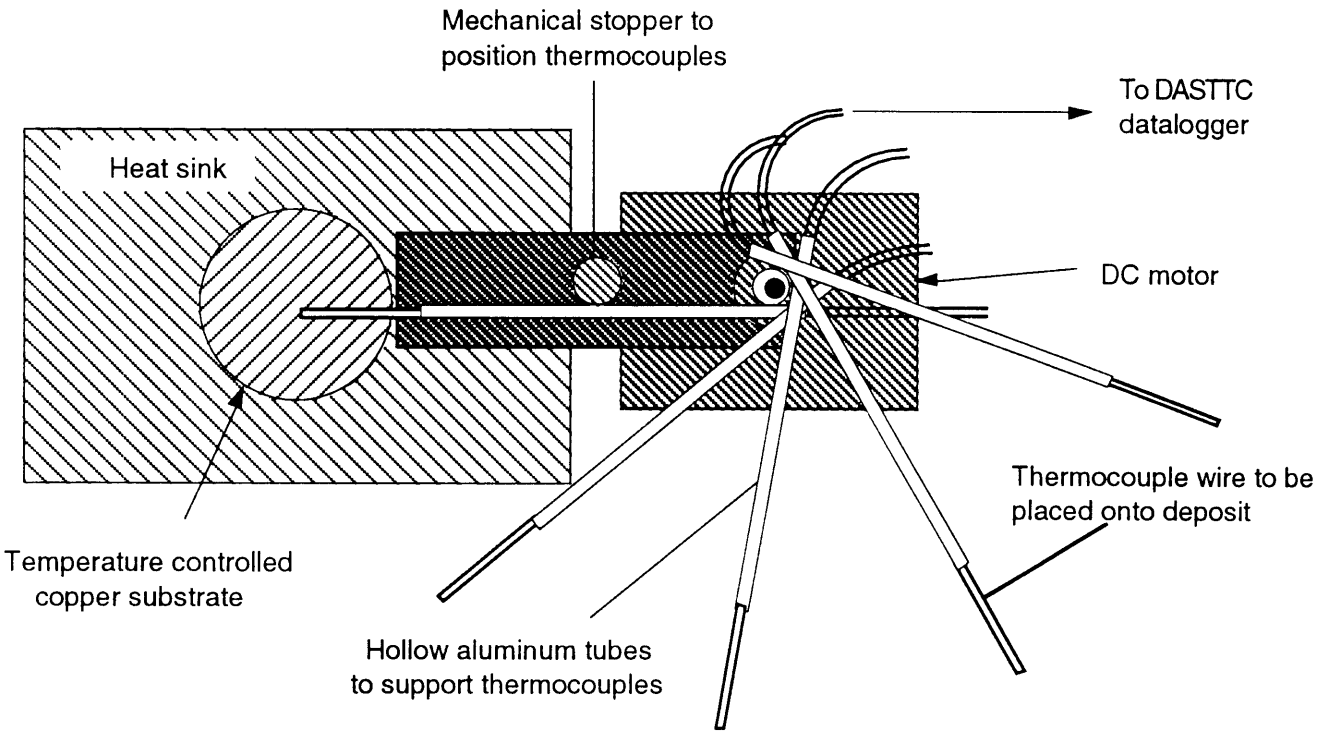


Fig. 9 Substrate setup before experiment began

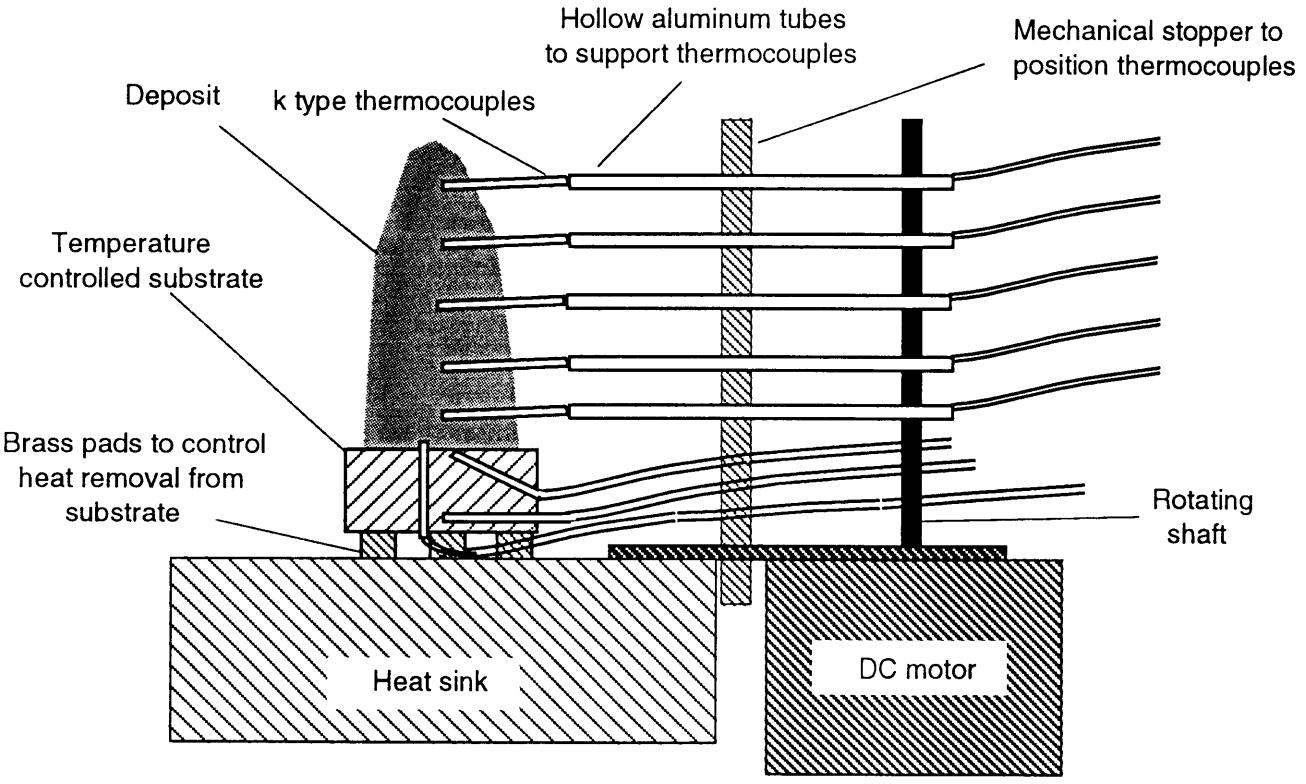


Fig. 10. Substrate setup after experiment finished

Two thermocouples were imbedded at the substrate surface to measure the temperature drop across the substrate/deposit interface. One thermocouple was placed adjacent to the heaters to provide feedback to the substrate temperature controller. The substrate was heated by two 75 W cartridge-type resistance heaters, embedded 20 mm below the top surface. During the experiment, thermocouples were placed onto the deposit surface at 5 mm, 9 mm, 28 mm, and 38 mm from the substrate surface to minimize the effect of the presence of thermocouples on the droplet spray and on the deposit thermal state. Since the experiment was performed in an enclosed chamber, a servo motor was employed to place the thermocouples into position one at a time. A photograph of the substrate taken after the experiment is shown in Fig. 11.

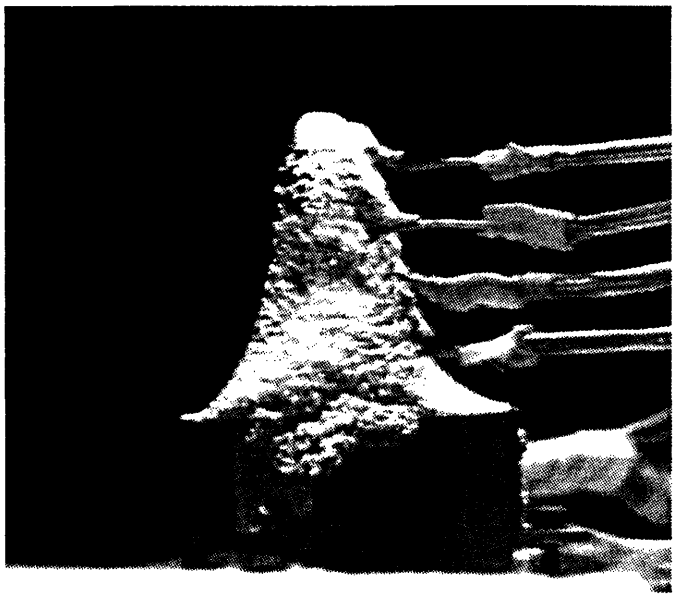


Fig. 11. Photograph of the deposit after the deposition experiment

Experiments were performed to directly measure the temperature of the deposit during spray deposition for verification of the deposit thermal model. A high liquid fraction was desired at the deposit surface, so that the thermal behavior throughout the solidification regime could be observed and the accuracy of the Scheil equation in this situation could be verified. This was accomplished by manipulating deposition rate and droplet thermal state.

The experimental conditions were designed using the droplet trajectory and thermal models. The experimental conditions are given in Table 1.

Table 1. Experimental conditions

Melt temperature in crucible	400°C
Pressure difference across orifice	10 kPa
Orifice diameter	125 μm
Distance from orifice to substrate	38 mm
Droplet charging voltage	650, 600, and 550 volts
Piezoelectric frequency	7.36 kHz
Substrate temperature, T_o	172°C

By providing multiple heating and cooling cycles, various thermal conditions were obtained for testing the deposit thermal model. To accomplish the objectives, the substrate was moved into the spray for 35 seconds and then withdrawn from the spray for 10 seconds. This cycle was repeated 18 times. Charging plate voltage was also adjusted during the experiment to provide a narrower spray, and therefore a higher deposition rate over a smaller area. This voltage was first decreased from 650 V to 600 V at 85 seconds into the experiment and then again to 550 V at 180 seconds into the experiment. The droplet trajectory simulation predicted the changes in spray radius for the three charging plate voltages used in the experiment as shown in Fig. 12. The deposit was built up to a thickness of 47 mm, with each deposition cycle increasing the thickness by 2.6 mm on average, reducing the droplet flight distance from 380 mm to 333 mm. The deposit thickness history shown in Fig. 13 reflects the effects of the spray radius changes and substrate motion.

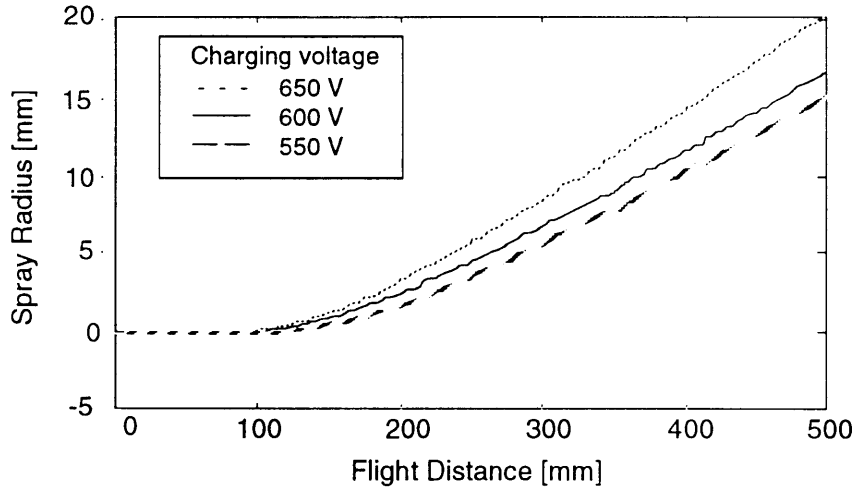


Fig. 12. Simulated spray radius vs. flight distance

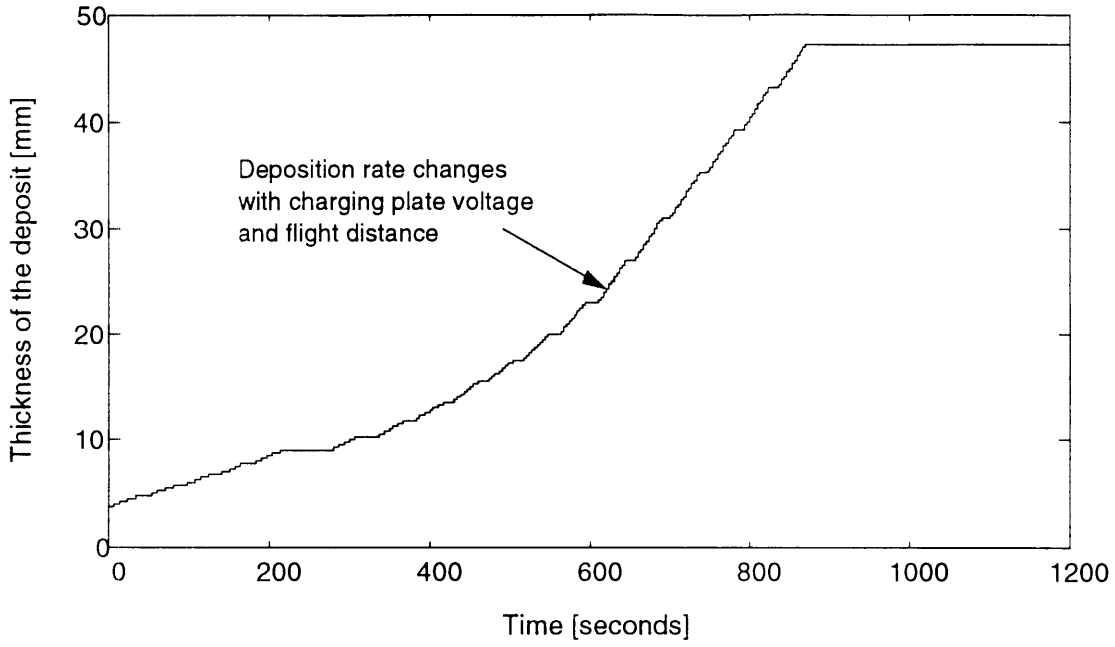


Fig. 13. Deposit thickness during the experiment

The thermal state of the impacting droplets changes with flight distance to the deposit surface as the deposit builds up. Flight distances from 380 mm to 333 mm provide droplets with a liquid fraction from 92% to 100%. Fig. 14 shows the droplet temperature history along the flight path.

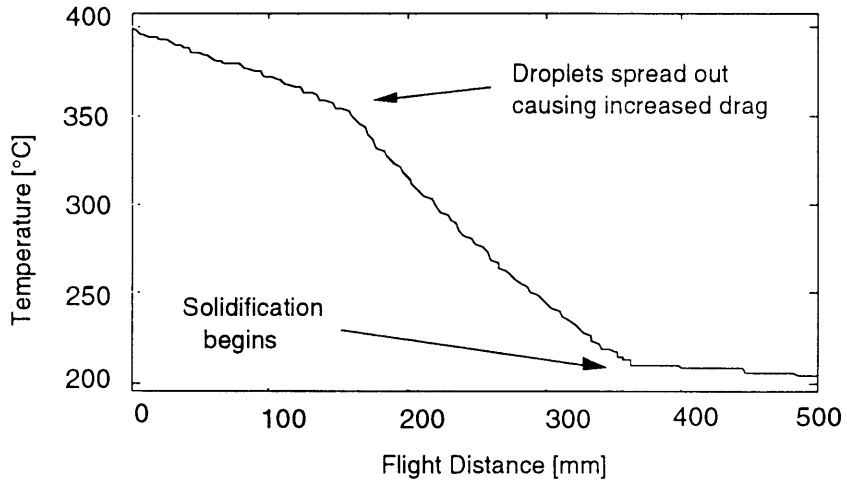


Fig. 14. Simulated droplet temperature vs. flight distance

3.2 Experimental results

Fig. 15 shows the measured temperatures of the deposit and substrate during the experiment.

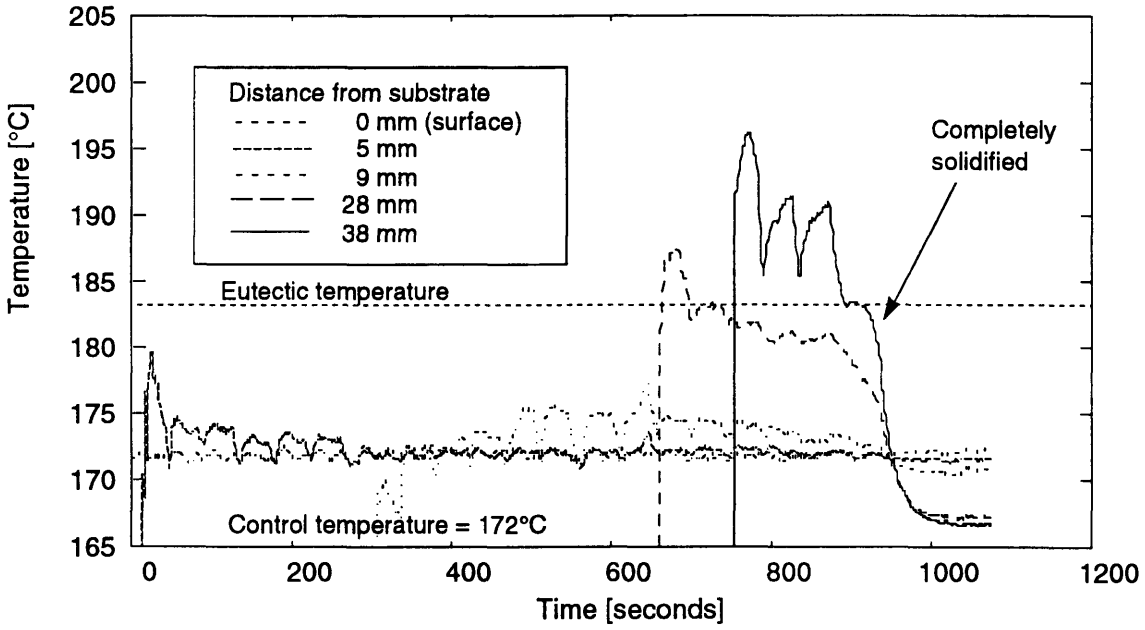


Fig. 15. Measured temperatures in experiment

Temperature measurements for each thermocouple are shown from the different points at which they were placed onto the surface of the deposit. In the first 400 seconds of the experiment, very little temperature rise above the substrate temperature was seen. Thereafter, an increase in deposition rate resulted in the continuous increase of the temperature at the deposit surface. At 700 seconds into the experiment, the eutectic temperature was detected 28 mm from the substrate, giving the first indication that liquid was present in the deposit. During the last three deposition cycles, large temperature peaks were observed at 38 mm from the substrate, revealing significant liquid fraction near the deposit surface. After the final deposition cycle at 870 seconds, temperature at 38 mm from the substrate cooled rapidly until it reached the eutectic temperature at 895 seconds. Once the liquid eutectic solidified, it continued to cool at a rapid rate. Throughout the experiment, the bottom of the substrate was held at 172°C. After deposition had finished,

the deposit surface cooled to 167°C, revealing the level of heat loss due to convection and radiation to the surrounding gas and chamber, respectively.

3.3 Simulation of experiment

The deposit thermal model was run for the specific experimental conditions outlined in Table 1. and the physical constants listed in Table 2. Both the deposition rate and droplet thermal state variations given in Figs. 14 and 15 are included in the deposit thermal model simulation. The algorithm employs an element thickness of 1.5 mm throughout the deposit except at the top element, which starts at an initial thickness of 0.5 mm and grows until the thickness is 2.0 mm, at which point it splits into two elements. The grid thickness in the substrate varies from a thickness of 1.5 mm at the surface element to a thickness of 10 mm at the bottom element.

Table 2. Physical constants of Sn-15 wt.%Pb

Melting temperature of Tin, T_M	232.2 °C
Liquidus temperature, T_ℓ	212.7 °C
Conductivity of solid, k_s	55.71 W/m°C
Conductivity of liquid, k_ℓ	27.81 W/m°C
Partition coefficient, K	0.096
Heat of fusion of Tin, $\Delta H_{f,1}$	56570 J/kg
Heat of fusion of Lead, $\Delta H_{f,2}$	26282 J/kg
Alloy specific heat, c_p	221.6 J/kg°C
Alloy density, ρ	7.957 kg/m^3

Simulation results are given in Figs. 16 through 18. The simulated temperatures at locations which correspond to the positions of the thermocouples during the experiment are shown in Fig. 16. The temperature predicted at 38 mm from the substrate is initially lower than the measured temperature. For subsequent deposition cycles, however, the predicted temperature is within 2°C of that from the experiment at the same location. Temperature predictions at a distance of 28 mm from the substrate are higher than in the experiment by approximately 5°C. At distances closer to the substrate, the simulated deposit temperatures are higher than the experimental temperatures at the corresponding locations by approximately 2°C.

Figs. 17 and 18 show constant temperature and liquid fraction contours resulting from the simulation. The top surface of the plot represents the deposit surface growing with time. Temperature variations are significantly decreased, showing a constant presence of

liquid in the deposit beyond 600 seconds into the simulation. Fig. 18 shows that liquid fraction begins building at 600 seconds and exceeds 45% before the simulation is complete.

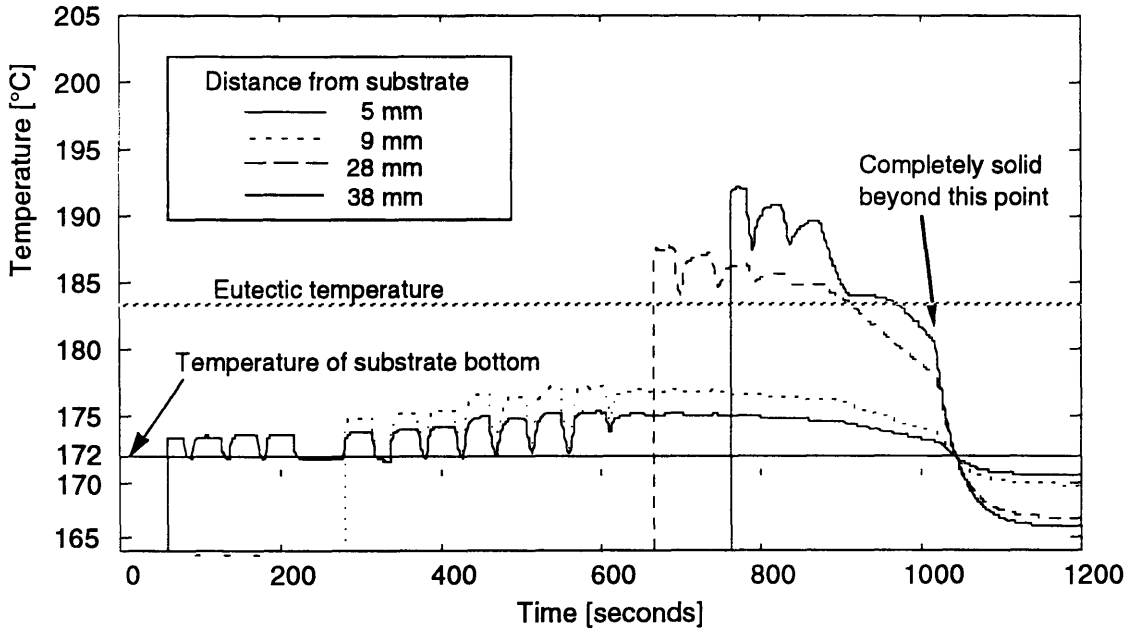


Fig. 16. Simulated deposit temperature at thermocouple locations within the deposit

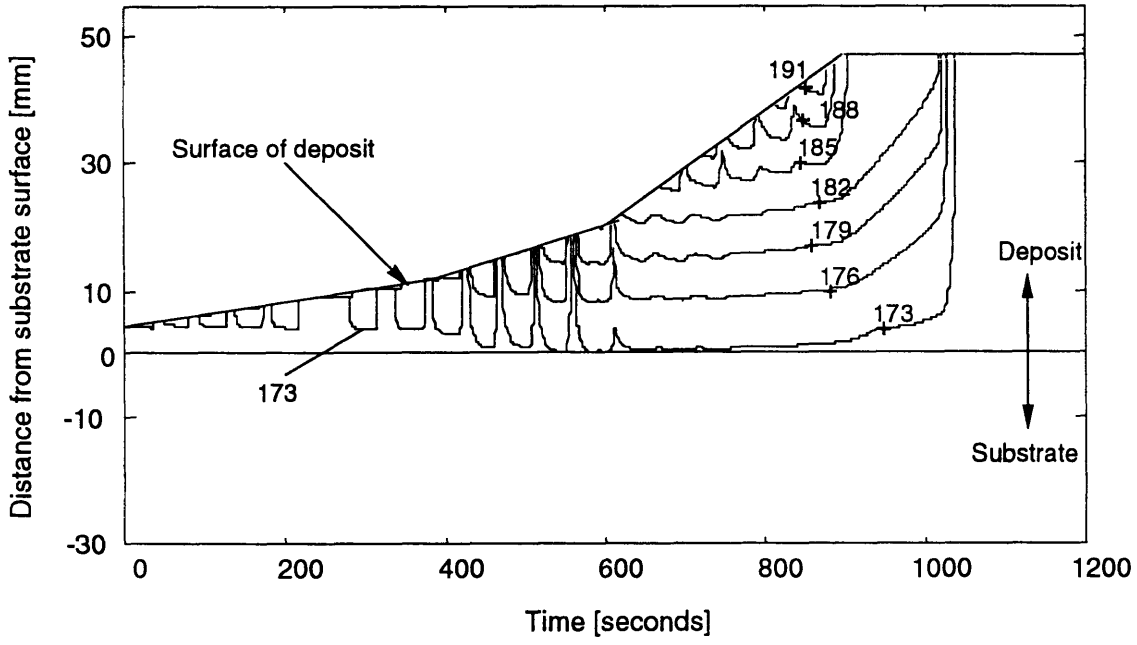


Fig. 17. Simulated temperature (constant temperature contours)

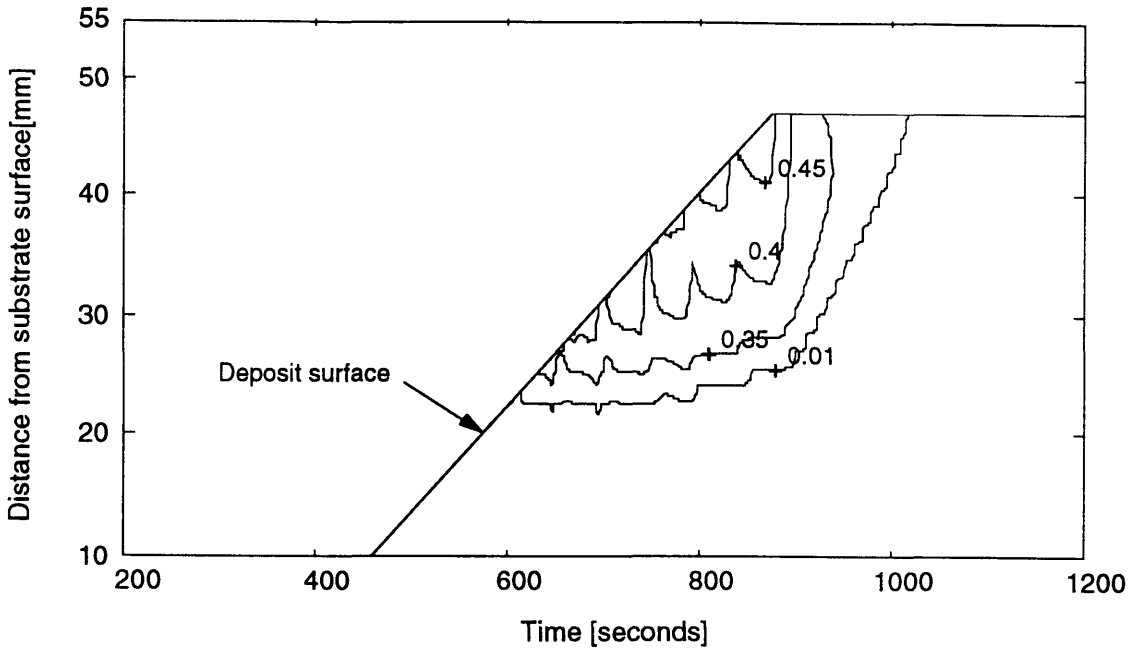


Fig. 18. Simulated liquid fraction (constant liquid fraction contours)

To determine the source of the discrepancy between the simulation results and the experimental data, the cooling rates within the deposit are compared after the deposit is completely solidified. This allows for a comparison between the simulation and the experiment without the effects of solidification. The rate of temperature drop after solidification is approximately 35% higher in the experiment than in the prediction. Also, the deposit is predicted to become completely solid at 1050 seconds compared to the experiment, which solidifies completely at 950 seconds. The additional 100 seconds needed for the simulation to solidify is due to the accumulation of liquid which occurs from a lower calculated heat extraction rate. The discrepancy in cooling rate can be explained by considering two-dimensional aspects of the deposit in the experiment.

3.4 Comparison of experiment to simulation

The one-dimensional model presented thus far neglects area variations along the z-axis as shown in Fig. 5. Increasing area closer to the substrate provides additional conduction paths into the substrate. In addition, the transverse radiation and convection from the lateral surfaces of the deposit are not accounted for. Since both of these phenomena would produce a lower temperature, the one-dimensional model can be considered an upper bound for the deposit temperature.

In order to estimate these effects, the deposit thermal model is modified to include area variations along the z-axis and radiation and convection on the side surfaces. Radiation and convection are added to each element through a source term. Because of the geometry of the deposit, a temperature gradient is expected in the transverse plane. Since the modified one-dimensional model still neglects any temperature gradient in the transverse direction, the predicted temperature is an average of the temperature profile. The thermocouples, on the other hand, are located along the z-axis in the deposit at the peak of the temperature distribution in the transverse plane; therefore, the predicted temperatures will be lower than the measured temperatures, constituting a lower bound on deposit temperature.

Fig. 19 shows the results of the one-dimensional model, the modified one-dimensional model, and the experimental data. As shown in Fig. 19, the experimental results are bound by these two simulations. As the deposit area is increased and the variation of area is decreased, the upper and lower bound models will converge to the actual results. Therefore, in the production of larger billets with uniform deposition areas, the deposit thermal model would produce accurate results to be used in design and process control.

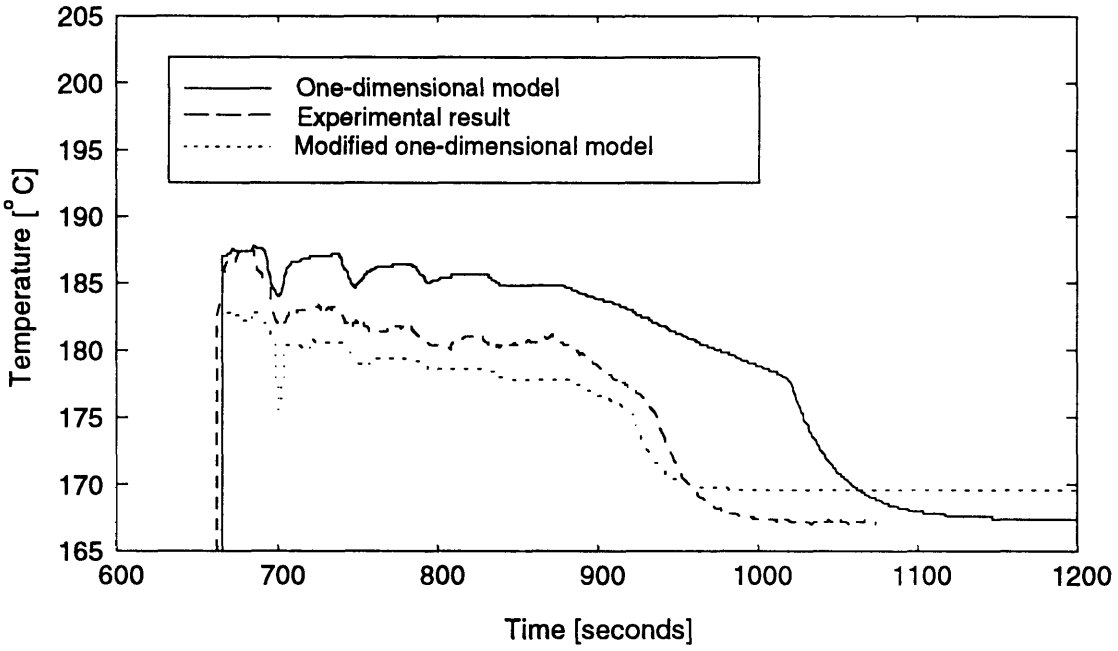


Fig. 19. Comparison of simulations Vs experiment at 28 mm.

3.5 Sensitivity analysis

Other factors which may have an effect on the simulation results include: 1) variations in mass flux due to mass flow and spray angle changes; 2) the assumption of the Scheil equation based solidification model; 3) the values of the thermo-physical properties used in the simulation; 4) the degree of thermal resistance at the deposit/substrate interface; and 5) the width or diameter of the deposit. The deposit thermal model can accommodate any necessary refinements in terms of the above aspects of the simulation. The deposit model was exercised at the experimental conditions to illustrate the impact of several assumptions changes on the computational results.

Mass flow rate

The deposit thermal state was determined to be very sensitive to changes in deposition rate. The simulation was run to show the impact of a change in deposition rate. The deposition rate was decreased by 20% and 50% at 600 seconds into the simulation to show the possible deposit thermal states which could have been generated in the experiment. Fig. 20 shows the predicted effect of mass flux change on the deposit thermal state. This sensitivity shows that by the judicious reduction of deposition rate, the deposit surface could be maintained below or at the eutectic temperature. Additional reductions in deposition rate would result in a completely solid deposit throughout the deposition. These results illustrate the importance of a stable and controllable spray for consistent mass flux. The sensitivity also demonstrates that by changing the mass flow rate on line, the thermal conditions of the deposit can be continually modified.

Solidification model assumptions

The initial simulation neglected the heat of mixing as well as the impact of non-equilibrium solidification. To quantify the impact of these assumptions, three cases were evaluated. Fig. 21 shows the temperature-enthalpy relationship for each of them. The first case applies the Scheil equation as described in chapter 2. The second case also uses the Scheil equation and in addition, includes the change in enthalpy caused by the heat of mixing. This was accomplished by adding the mixing term into the enthalpy as given by Poirier [22] for both the liquid and the solid. The enthalpy was then referenced back to 25°C (H = 0). Because the heat of mixing is higher in the solid phase than the liquid phase, the effective heat of fusion for the alloy is lower. For the third case, the effect of rapid solidification was qualitatively explored. When rapid solidification occurs, the solidification front moves faster than the rate of solute diffusion, resulting in an entrapment of solute which leads to a smaller percentage of eutectic compound.

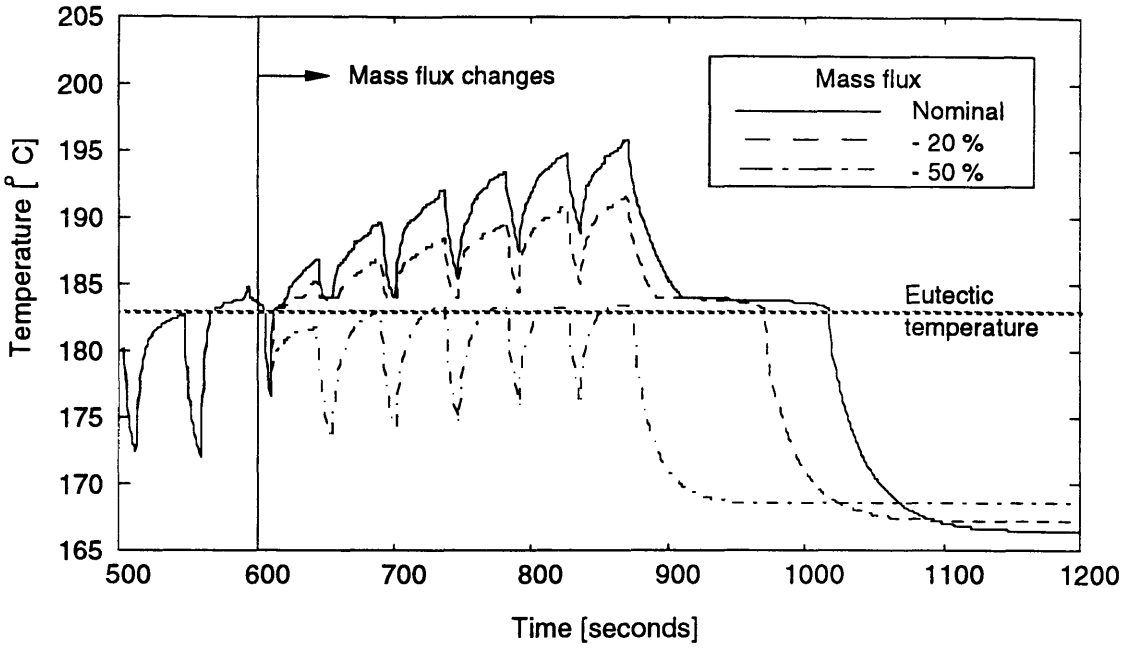


Fig. 20. Effect of mass flux changes on the deposit surface temperature

The dynamics of rapid solidification are different for every condition and cannot be adequately modeled with an explicit temperature-enthalpy relationship. However, a straight line assumption was chosen to capture the impact qualitatively. As with a rapidly

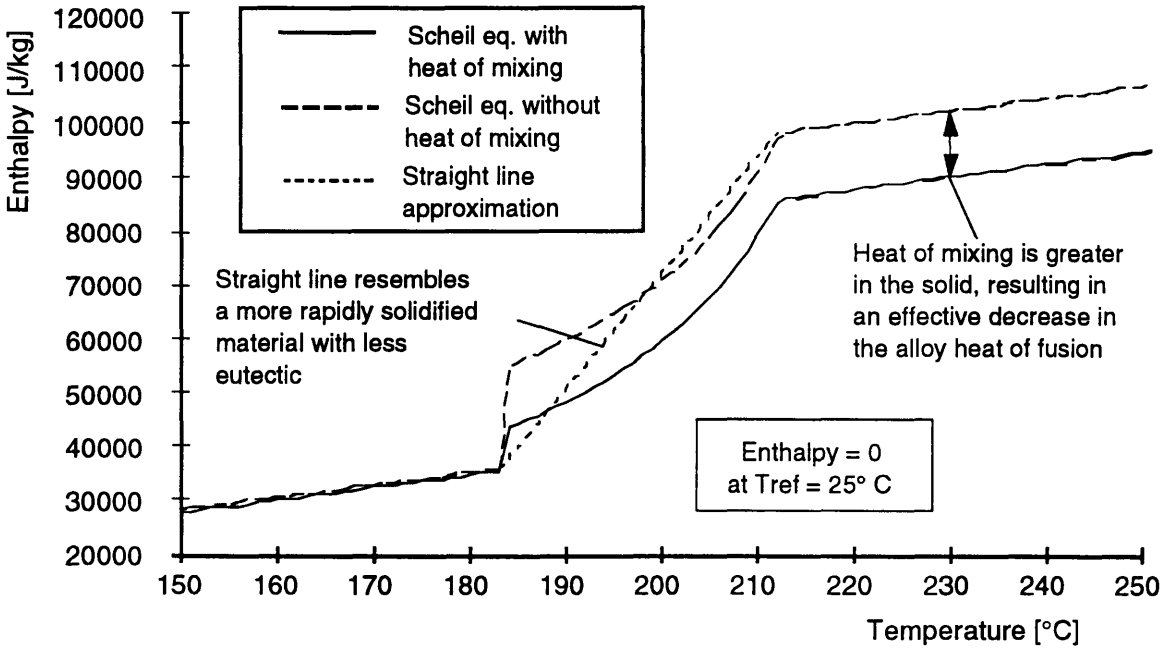


Fig. 21. Temperature-enthalpy relationships

solidified material, this relationship assumes no eutectic material is present throughout solidification.

Fig. 22. shows the simulated deposit surface temperature for the three solidification cases. As expected, the addition of heat of mixing decreased the temperature of the deposit and reduced the amount of liquid remaining at the end of deposition.

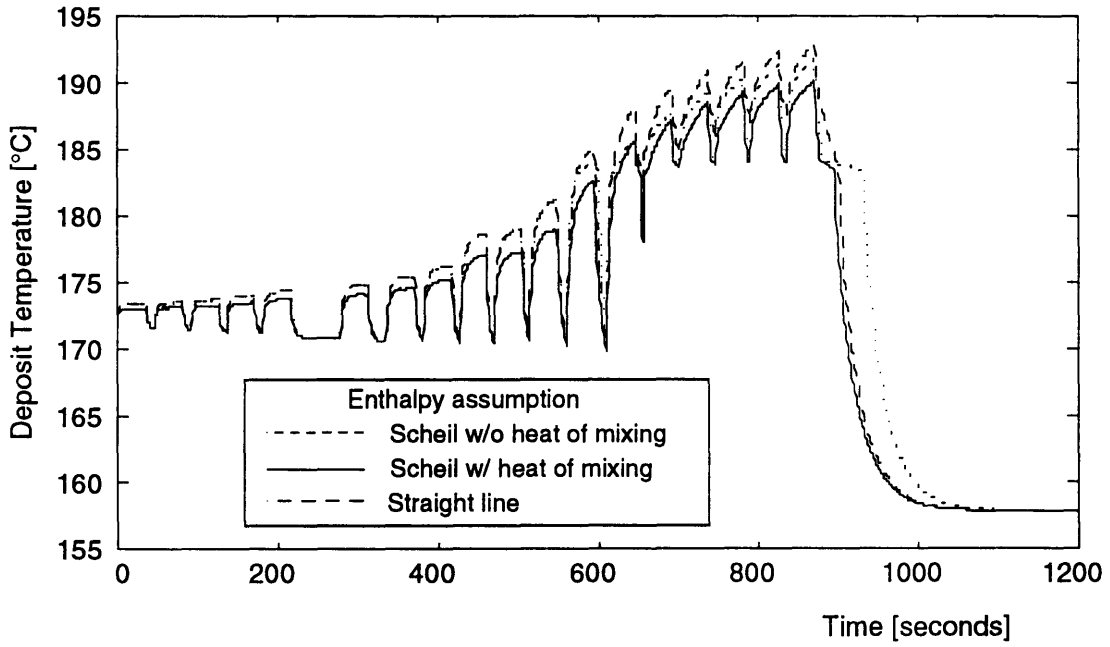


Fig. 22. Effect of solidification assumptions on simulation results of the experiment

As a result, the deposit in the second case completely solidifies ~50 seconds before that in the first case. The third case, employing a straight line assumption which did not account for heat of mixing, was also cooler than the baseline case, but for different reasons. The straight line assumption causes more deposit to solidify at higher temperatures. As the deposit is maintained at these higher temperatures, the driving thermal diffusive force becomes greater, resulting in an increased cooling rate. At conditions in which the solidification model is important, the deposit temperature is significantly high to limit the amount of rapid cooling; therefore, given the relatively small effect on the final solution, it appears that rapid solidification can be neglected on the macroscopic scale. The heat of mixing, however, is significant and should be accounted for provided the data for the material is available.

Deposit contact heat transfer coefficient

The phenomenon which determines the contact coefficient is difficult to account for in this complex process. Thermal resistance is created by porosity or roughness between two surfaces. Contact coefficient is usually of concern when two solid materials are brought into contact with an applied pressure. A review of the literature reveals that most values of contact coefficients in the described situation fall between 1,000 $W / m^2 °C$ and 30,000 $W / m^2 °C$ [23]. Since the droplets in spray forming are mostly liquid upon impact, the deposited material can flow into microscopic gaps, making the contact better than in the cases of two solids in contact.

A problem sometimes experienced in spray forming is that the deposit shrinks as it cools, causing microshrinkage and gaps at the interface. These gaps cause an increase in thermal resistance at the interface. In extreme cases, the deposit can contract and curl upward upon cooling, resulting in a significant gap at the interface. However, since the highest deposit temperature experienced in this study was approximately 200°C and the lowest temperature experienced in the deposition was 173°C, the amount of shrinkage is small. As a result, it is safe to assume that the value of the contact coefficient is at the high end or beyond the range experienced for two solids in contact.

A sensitivity analysis was performed to determine at what level of contact coefficient appreciable changes in the temperature field are observed. Fig. 23 illustrates that the temperature field starts changing significantly at a contact coefficient below 2000 $W / m^2 °C$. In light of the published data and the observed temperatures in the experiment, the contact resistance at the deposit/substrate interface can be neglected.

Deposit width

Since the model is one dimensional, a key assumption is that conduction and convection through the sides of the deposit are negligible. The comparison made in this chapter between experimental and simulation clearly show the impact of the geometry of the deposit. The model was run two different ways to bound the thermal solution. The deposit thermal model assumes that the temperature is constant for a given ζ (Fig. 5). The convection added to the side of each element is assumed to be equal to that of the top convection. In an attempt to determine at what diameter the two dimensional effects are negligible, several different deposit diameters were applied to the model. The effects of transverse convection and conduction can be neglected for geometries that show that the two methods of solving result in a similar thermal solution.

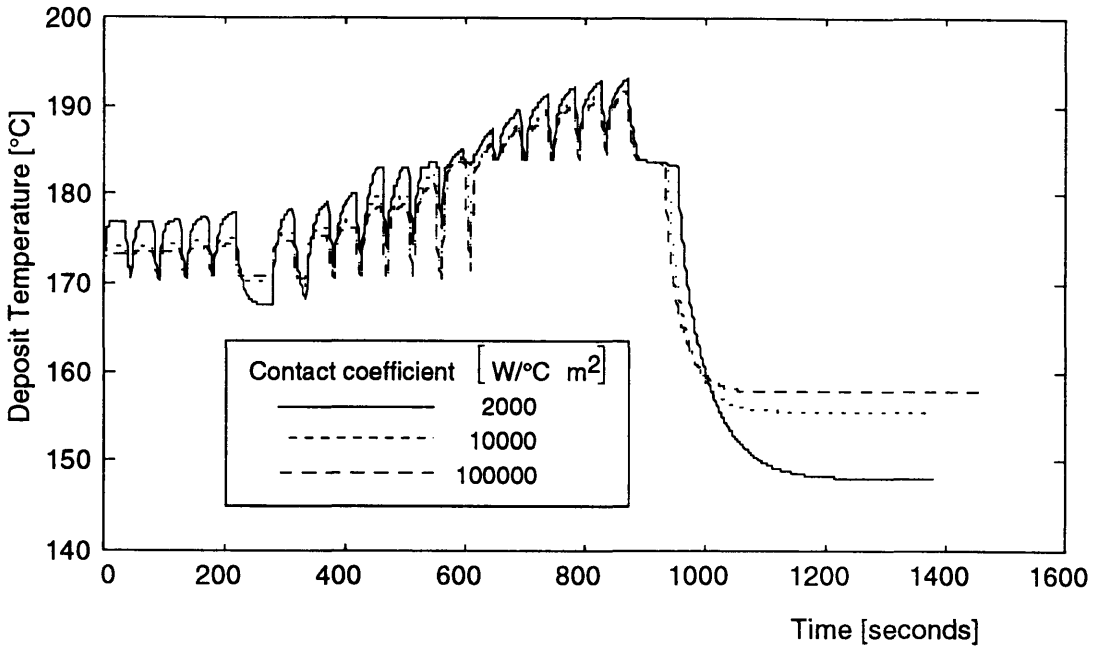


Fig. 23. Effect of contact coefficient on simulation results of the experiment

Fig. 24 shows the results of the simulation for three different deposit diameters. Once the deposit width reaches 200 mm, the effect of transverse convection on the temperature solution begins to diminish. It is therefore important to note that the experiments performed will result in a temperature less than that of the ideal one dimensional case. To make a generalized comparison, we consider the aspect ratio of the deposit, which is defined as the deposit height divided by the deposit width. This study used a specific deposit height which totaled 47 mm. Therefore the critical aspect ratio of the deposit is 0.25 at the calculated critical width of 200 mm. It should be noted that the model neglects any gradient in the transverse direction and therefore these results should serve as only a guide.

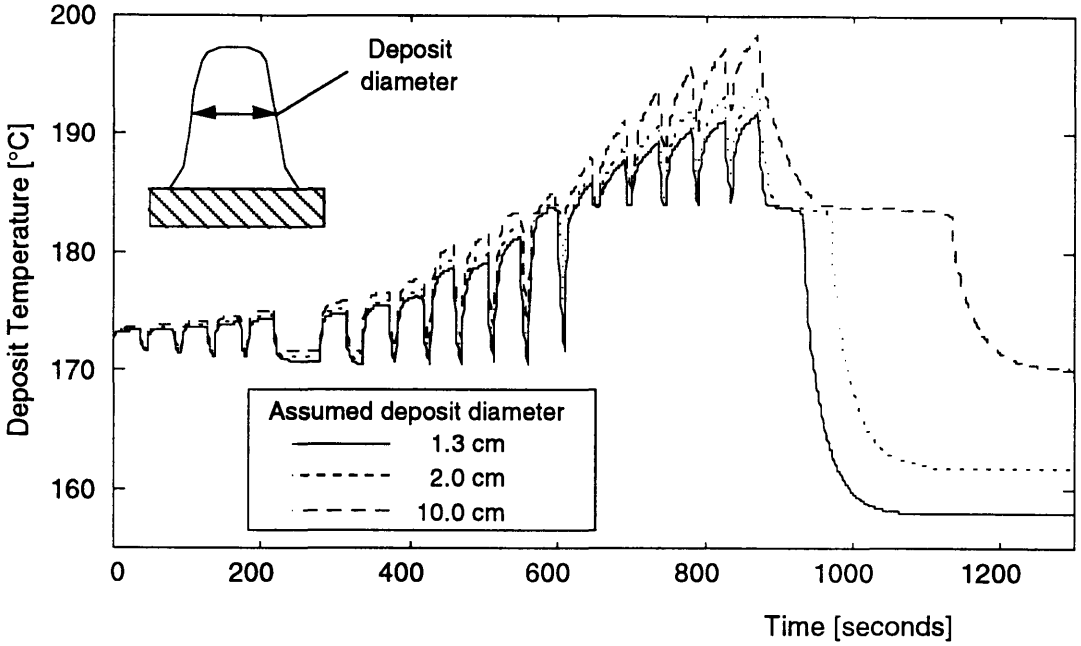


Fig. 24. Effect of deposit diameter on solidification on simulation results of the experiment

Chapter 4

MICROMODELING OF DROPLET SPLATS

4.1 Introduction

The simulations presented thus far have treated the spray as continuous and uniform at the deposit surface. This approximation is completely valid when looking at the macroscopic, or 'time average,' thermal state of the deposit. However, by modeling the deposit in this way, the initial rapid cooling and solidification which occurs in the droplets upon and after impact is not explained. In cases where the deposit surface temperature is lower than the melting temperature, the solidification will occur completely on the droplet scale. For this situation, it is necessary to look at the solidification of the individual droplet splat.

4.2 Assumptions

In order to utilize the deposit thermal model from chapter 2, several additional assumptions had to be made. They are listed here and will be treated in more detail below:

1) The droplet splat solidification time is much longer than the spreading time.
2) The thermal resistance between the splat and the deposit surface is chosen to be $1 \times 10^7 \, W / m^2 °C$.
3) The droplet splat becomes equal to the deposit surface temperature before the next droplet arrives.
4) The splat solidification follows local equilibrium solidification.

1) To justify the first assumption, the results of a computational droplet study are compared to calculated cooling times. As reported by Dykhuizen in a review paper on droplet impact and solidification research [24], scaling arguments and experimental evidence have indicated that the solidification time is typically two orders of magnitude longer than the splat flow time. Computational studies performed by Liu and Dandy [25] show that for similar size and composition, the droplets completed spreading in approximately 0.015 msec. The following results will show that solidification occurs over a significantly larger time period, justifying the treatment of splat solidification as pure conduction in a stationary splat.

2) To evaluate the influence of thermal resistance at the droplet interface, a 25 μm splat thickness, ζ is considered. The resistance within the droplet due to conduction through the liquid is approximated by:

$$h_{eq} = k / \zeta \tag{15}$$

where h_{eq} is the equivalent conductance through the droplet and ζ is the droplet thickness. The value of h_{eq} is 1.2 x 10^6 $W / m^2 °C$ for the experimental conditions described in chapter 3. Typical values reported range from 1 x 10^4 $W / m^2 °C$ to 1 x 10^6 $W / m^2 °C$ [26-29]. Provided that the reported values of contact coefficients are accurate, this would suggest that the contact coefficient has an impact, since it is of the same magnitude as the equivalent thermal resistance of the droplet splat. However, choosing a value on the high end of the reported data would have less an effect. Since there is no rigorous justification for applying a particular contact coefficient, the value was assumed to be sufficiently high so that the contact resistance would be negligible. The contact coefficient value was assumed to be 1 x 10^7 $W / m^2 °C$. This approach also puts an upper bound on the rate of cooling and solidification within the droplets.

3) For the third assumption to be justified, the solidification time and frequency of impact must be considered. A typical experiment creates 50,000 droplets per second or 1 droplet every 20 μsec. Assuming that the spray is 20 mm in diameter and that the droplets spread evenly through the spray, a droplet will land at the same location approximately every 20 msec. The analysis will show that this assumption holds for nearly all the droplet conditions in the experiment. At the deposit surface temperatures above the eutectic, the droplet solidification cannot be complete although the initial solidification above the eutectic still occurs within 20 msec.

4) The assumption of local solidification is the hardest to justify. The real case will undoubtedly have rapid solidification which can only be modeled with a front tracking model with a thermo-kinetic model which relates undercooling to interface velocity. However, for illustrating the usefulness of the model at hand, local solidification will be assumed. For future studies, the front-tracking algorithm included in the deposit thermal model, which will not be addressed further here, can be utilized.

The droplet splat thickness was determined by comparing reported splatting geometries from both experimental and computational research. The flattening ratio is given as a function of Reynolds number, which is defined as:

$$Re_d = \frac{\rho V D_d}{\mu} \tag{16}$$

where Re_d is the Reynolds number, V is the droplet velocity, D_d is the droplet diameter, and μ is the viscosity of the alloy. In the experiment, the droplet diameters were approximately 236 μm and the impact velocity was 2.76 m/sec, resulting in a Reynolds number of 2,500. Several correlations have been published which estimate the flattening ratio to range from 2.5 to 6.0 for this Reynolds number [30]. Using Jone's relationship [26], a flattening ratio of 2.5 was calculated, resulting in a splat diameter of 586 μm. For a cylindrical splat, the thickness is 25 μm.

4.3 Computational considerations

To adopt the deposit model to the droplet scale, several enhancements were made. In the model the elements were reformulated to allow for a contact resistance between the splat and the preexisting deposit. The deposit and droplet both used the same solidification model of the macroscopic model. To run the droplet case, a row of elements is placed upon the deposit elements at the temperature of the droplet upon impact. The difference between the macroscopic model and the droplet splat, or 'microscopic model,' is the size of the elements and the time over which the simulation occurs.

4.4 Simulation results

Experimental conditions were duplicated for three deposit surface temperatures to address several droplet splat conditions which occurred over the length of the experiment. Figs. 25 and 26 show the temperature at five different locations within the droplet. Fig. 25 shows the simulation for a deposit surface temperature of 153°C. The model predicts that the droplet will completely solidify in 4 msec. In contrast, Fig. 26 shows the same plot for a surface temperature of 182°C. In this operating environment, the droplet continues to solidify beyond 12 msec.

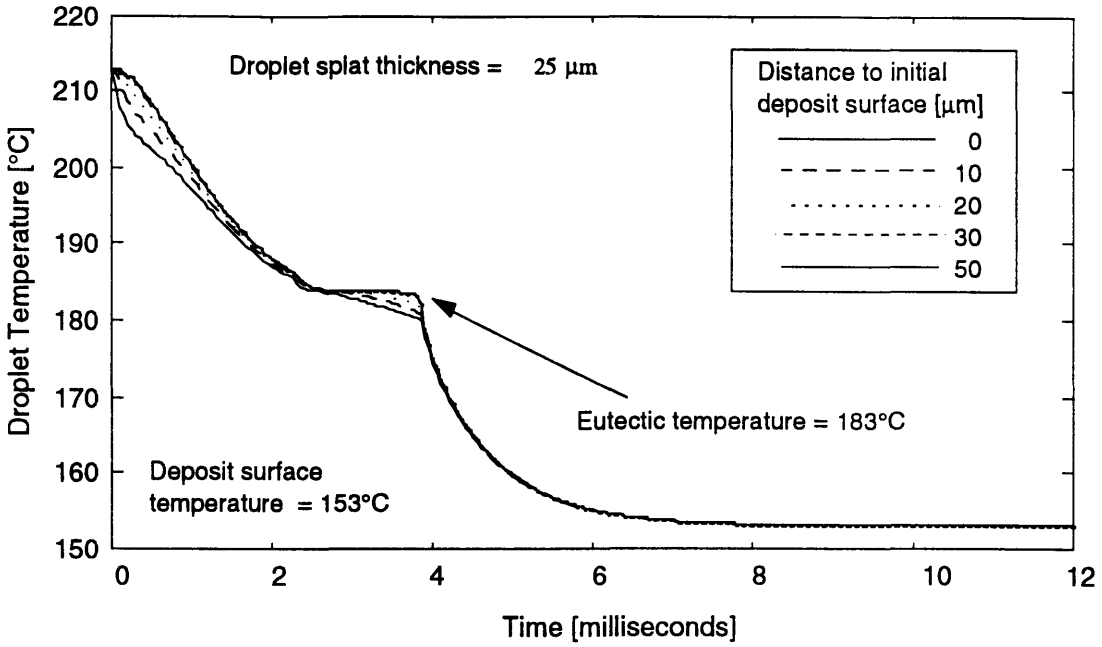


Fig. 25. Droplet solidification at a deposit temperature of 153°C

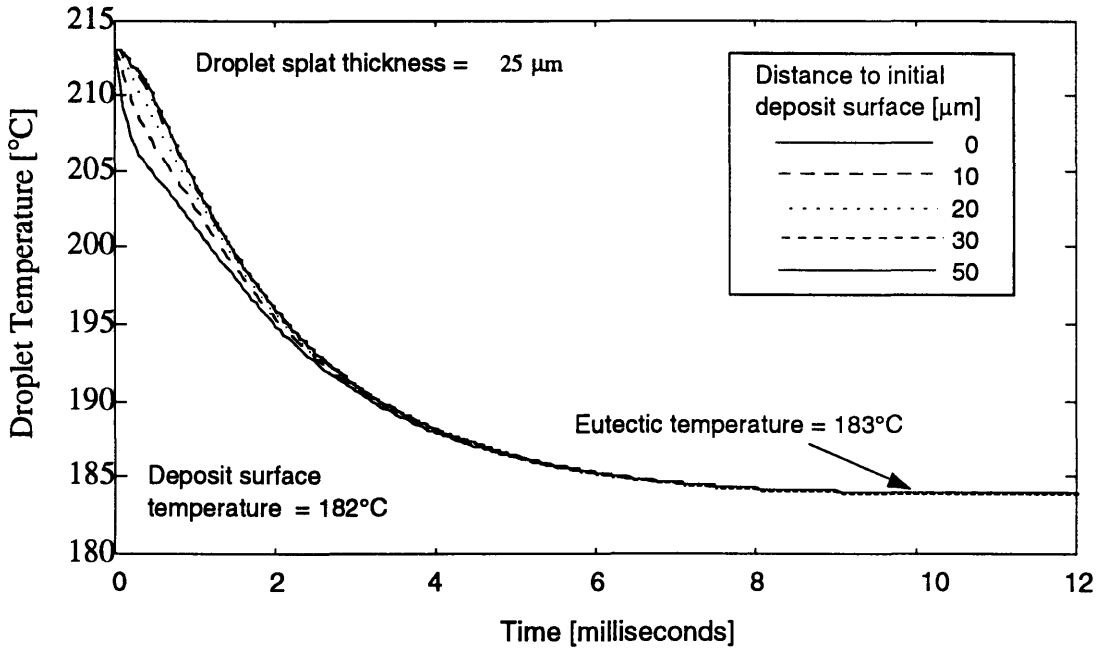


Fig. 26. Droplet solidification at a deposit temperature of 182°C

Fig. 27 compares the droplet splat surface temperature for four different deposit surface conditions. The figure shows that even with relatively hot deposit surfaces, the solidification rate above the eutectic does not change significantly.

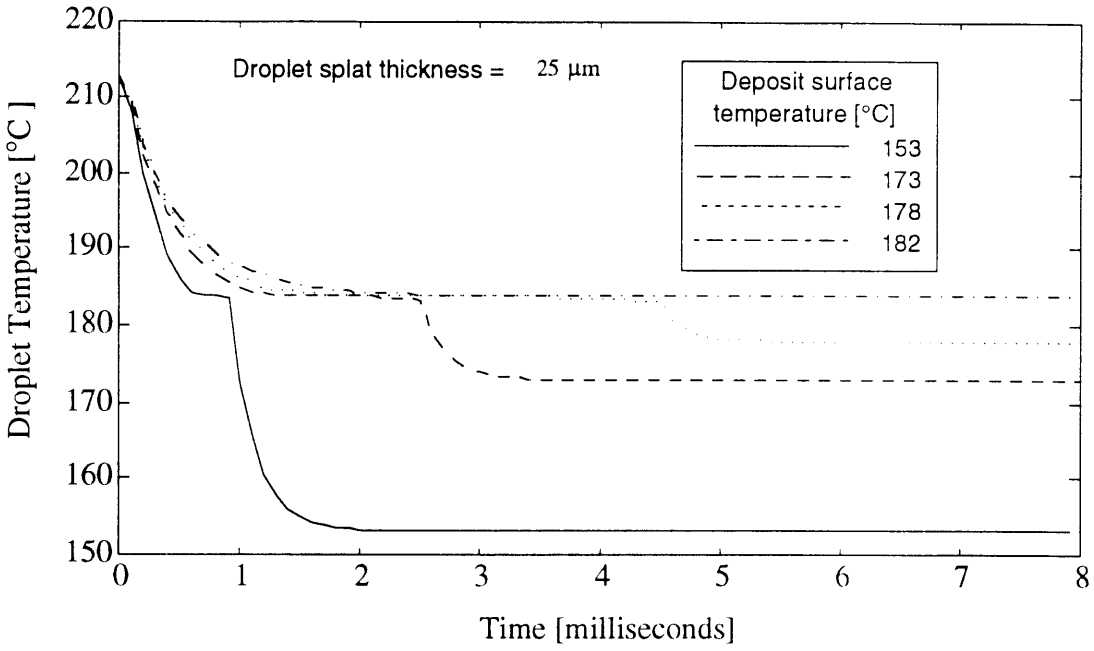


Fig. 27. Droplet surface temperature at different deposit conditions

4.5 Discussion

Many attempts have been made to characterize droplet microstructure in terms of solidification rate, however, less attention has been given to the characterization of microstructure in the droplet splat in terms of solidification upon impact. The simulations exercised here offer a starting point for understanding the droplet splat phase of microstructure development within the UDS process. These most significant aspect of these results is that even with a deposit temperature at the eutectic point, rapid cooling at the beginning of the splat cooling is present. The major influence of the deposit temperature is on the extent of solidification which occurs rapidly during rapid splat cooling.

With the modeling of splat cooling on the microscale, an understanding of several stages of cooling has now been gained. The thermal history and microstructural evolution through the entire process is summarized below into three stage of cooling and solidification:

1) Droplet flight - The droplet is cooled by convection to the surrounding gas and radiation to the surrounding chamber as it is propelled to the substrate surface. Nucleation can occur during the droplet flight as the droplet temperature reaches some critical undercooling. Upon nucleation, a brief period of rapid solidification will occur. The extent of this brief period of solidification is a function of the amount of undercooling present at the time of nucleation. As solidification occurs, latent heat is released, causing the droplet temperature to

rise until it reaches the liquidus temperature. This phenomenon is commonly referred to as recalescence. Once the liquidus temperature is reached, a period of slower cooling continues while the droplet is still in flight. The rate of solidification is then driven by the heat extraction from the droplet until impact is made with the deposit.

Before the next stage of solidification begins, the droplet splats upon impact to the deposit. The droplet, whether fully molten, partially solid, or completely solid, impacts the surface, causing the droplet to splat and deform. In the case where solid exists in the droplet, fragmentation of the already solidified dendrites occurs, producing nucleation sites throughout the splat. Similarly, if the deposit surface is partially liquid, fragmentation of the deposit dendrites may also occur. In addition, liquid from the droplet or the deposit surface will help to fill any voids which may initially exist when a substantially solid droplet impacts. The amount of spreading will also influence the following stage of cooling since a wider splat will create more contact with the cooler deposit.

2) Droplet splat rapid cooling - During and after droplet spreading, a second period of rapid cooling occurs. The rate and extent of rapid solidification is driven by the initial temperature difference between the droplet splat and the deposit surface. This case is different from the rapid cooling in the droplet because there is a significantly stronger heat sink which can be considered infinite in comparison to the droplet. Therefore, the rapid cooling period will continue until the droplet splat reaches the deposit surface temperature.
3) Deposit slow cooling - Once the droplet splat reaches the deposit surface temperature, the splat can be considered consolidated into the deposit. At this point and beyond, there is further solidification and cooling which occurs at a much slower pace. This rate is driven by the macroscopic thermal evolution of the deposit. The microstructure can then evolve through coarsening and ripening mechanisms while the deposit is maintained at elevated temperatures.

Depending upon how process parameters are manipulated, the percentage of solidification within each of these stages can be adjusted. For example, if fully molten droplets are supplied, the first stage of solidification is completely eliminated. Further manipulation can occur through the deposit temperature. At a deposit temperatures below the eutectic, the solidification occurs rapidly within the droplet splat. At elevated deposit temperatures, a larger portion of the solidification occurs slowly as the deposit cools "macroscopically". By understanding how these individual portions contribute to the

evolution of microstructure, the process can be manipulated to obtain the desired final microstructure.

Chapter 5

DEPOSIT MICROSTRUCTURE

5.1 Introduction

Up to now, this research has concentrated on the thermal state evolution of the droplets and deposit. However, prediction of thermal state is ultimately needed to draw correlations with the resulting microstructure, providing a connection between process parameters and microstucture. With the thermal modeling of the entire process in place, correlation to microstructure can now begin.

Microstructure can be split into two components: morphology and composition. The morphology refers to the shape and size of the grains or crystal structure within the material. The composition refers to the amount of solute in the primary and secondary phases of the material as well as the extent of segregation. Morphology is believed to be driven by two phenomena. The first is the mechanical fragmentation which occurs at the point of droplet splatting [31]. The second is the thermal gradient and solidification rate which naturally selects a morphology based on the minimization of energy [32]. The composition on the other hand, is driven by the rate of solidification and the overall time at elevated temperatures. At slow rates of solidification, solute distribution follows the equilibrium phase diagram, which is defined by the physical constants listed in Table 2. During rapid cooling conditions, deviations from equilibrium occur, resulting in less solute rejection and higher percentages of solute.

The experiment in chapter 3 provides an excellent opportunity to understand the microstructural evolution within the deposit. The experimental parameters were set such that there was only a small fraction of solid within the droplets upon impact. Therefore, a majority of the solidification and microstructural evolution occurred within the deposit. Due to the adjustments made throughout the experiment, the deposit experienced a wide range of temperatures. It is therefore possible to see the microstructural impact of different thermal histories within a single experiment.

The microstructure deposit samples from the experiment were examined under an SEM (model: JEOL Superprobe 733). Micrographs of the microstructure were taken and a composition analysis through the deposit was performed.

5.2 SEM sample preparation

The Sn15wt.%Pb deposit was removed from the substrate intact. A transverse cross-sectional slice, perpendicular to the substrate surface, was cut from the deposit and then cut in two, creating a top and bottom sample. The two samples were mounted in clear-cast epoxy and polished to a final grit size of 0.3μm. A solution of 2 vol.% HCl - 5 vol.% HNO_3 in methanol was used to enhance the microstructures.

5.3 SEM micrograph results

Photographs were generated using the back scattered electron technique. Fig. 28 shows a summary of the microstructure at different distances from the substrate.

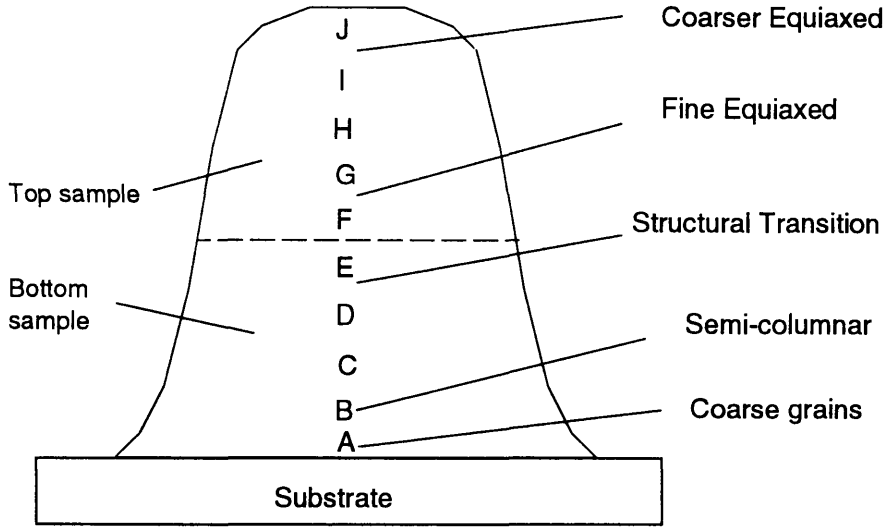


Fig. 28. Schematic of deposition microstructure after experiment

Fig. 29 illustrates the dramatically different microstructures which can be created when the deposit experiences different deposit temperatures. The bottom of Fig. 29 shows a thin layer of material that was reheated to 200°C to ensure good contact with the substrate. This allowed for extensive coarsening to occur. The microstructure directly above this interface is indicative of rapid solidification and shows fine precipitates dispersed evenly throughout. Figs. 30 and 31 also reveal a fine uniform microstructure with evenly dispersed precipitates. As the distance from the substrate increased, the coarseness of the microstructure also increased. At 24 mm from the substrate, the amount of precipitates increase at the grain boundaries to a point at which the grains are distinguishable. Figs. 32 through 34 show that the grain size continues to grow with distance from the substrate. At 38 mm from the substrate, eutectic structure is more abundant, signifying that the solidification occurred at close to equilibrium conditions.

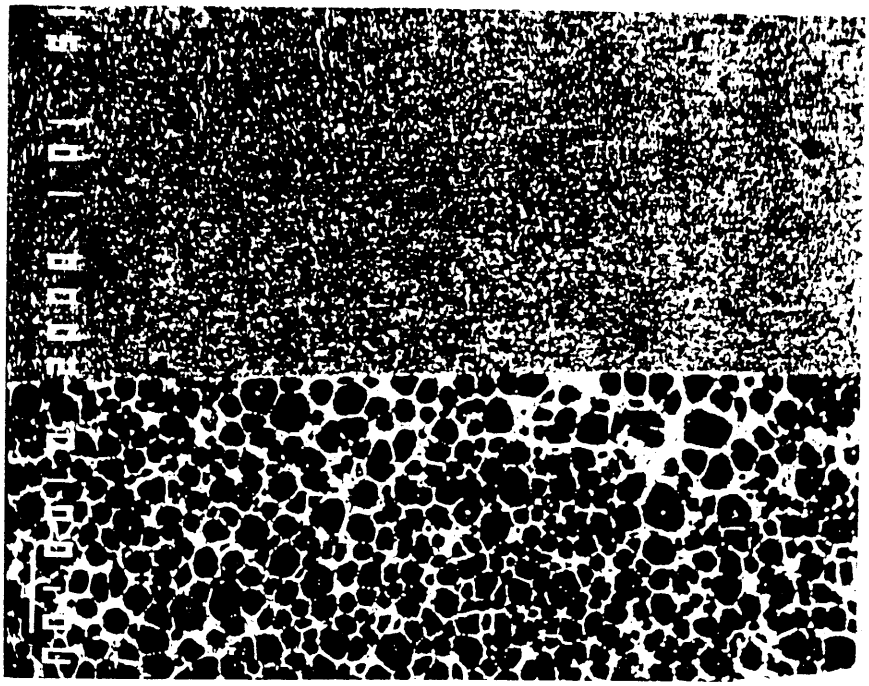


Fig. 29. A SEM photograph of Sn-15%wt.Pb alloy at a transition in microstructure produced by remelting a thin layer of alloy at a temperature of 200°C and then depositing liquid droplets while the deposit temperature was at ~175°C (pos. A & B in Fig. 28).

Fig. 30. A SEM photograph of Sn-15%wt.Pb alloy with fine equiaxed microstructure produced by depositing partially solid droplets onto a 175°C deposit (pos. C in Fig. 28).

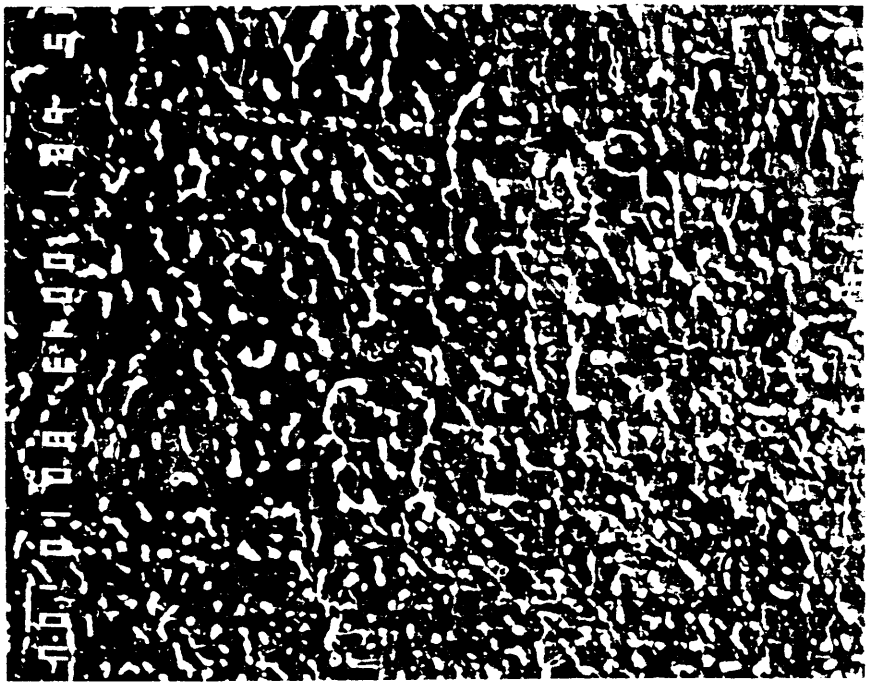


Fig. 31. A SEM photograph of Sn-15%wt.Pb alloy fine with equiaxed microstructure produced by depositing partially solid droplets onto a 175°C deposit (pos. C in Fig. 28).

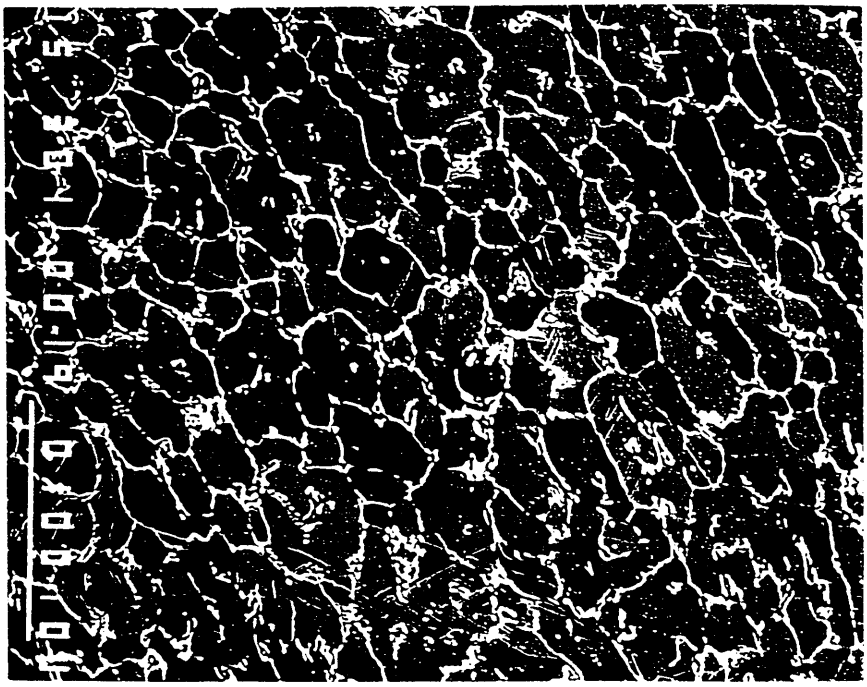


Fig. 32. A SEM photograph of Sn-15%wt.Pb alloy with equiaxed microstructure produced by depositing partially solid droplets onto a 180°C deposit (pos. F in Fig. 28).

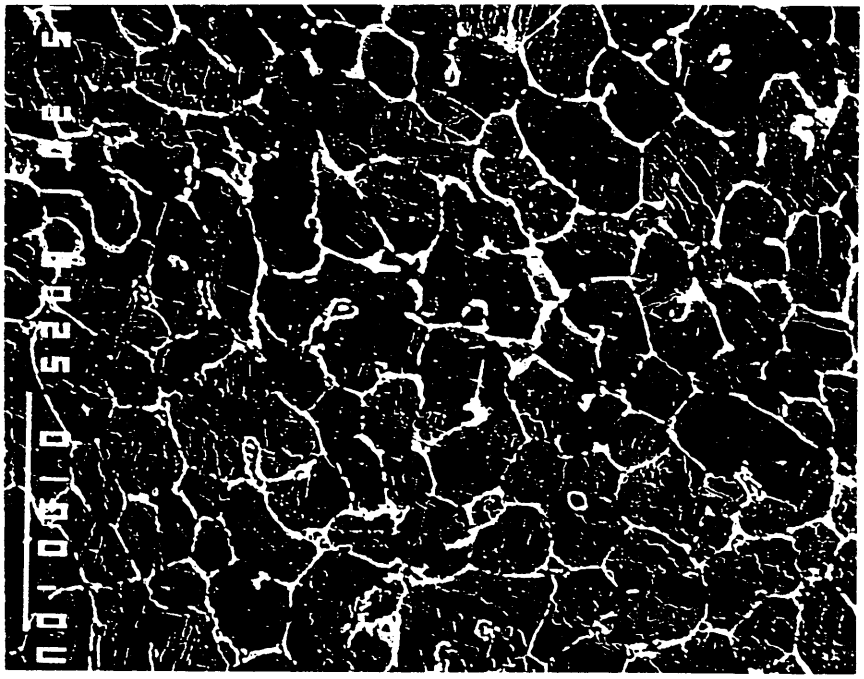


Fig. 33. A SEM photograph of Sn-15%wt.Pb alloy with equiaxed microstructure produced by depositing partially solid droplets onto a 183°C deposit (pos. H in Fig. 28).

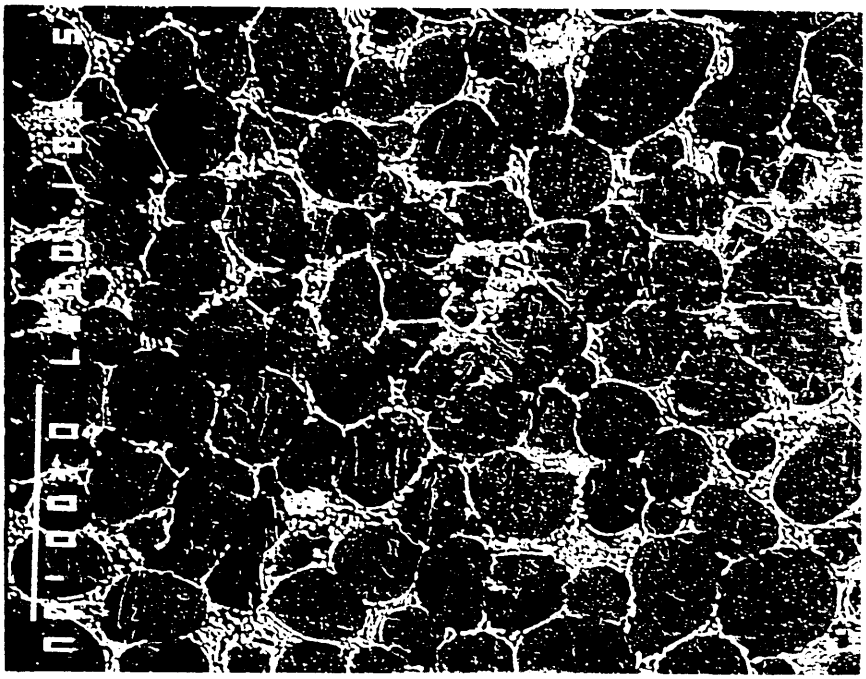


Fig. 34. A SEM photograph of Sn-15%wt.Pb alloy with course equiaxed microstructure produced by depositing partially solid droplets onto a 190°C deposit (pos. J in Fig. 28).

5.4 Compositional variation

To further understand the microstructure within the deposit, compositional analyses were performed using both energy dispersive spectrum (EDS) analysis. The intent of the analysis is to determine the extent of non-equilibrium solidification by comparing the actual solute levels to equilibrium values. The sensitivity studies from chapter 4 show that the temperature field can vary greatly depending on the extent of solute in the liquid phase.

Table 3 shows the composition of solute in each phase as well as the percentage of each phase at locations throughout the deposit as given in Fig. 28.

Table 3 - Compositional variations of deposit per SEM analysis

Location	Distance from Substrate [mm]	%Pb in Sn rich	%Sn in Pb rich
A	1	2.50%	0.25%
B	2	4.10%	0.41%
C	5	3.60%	0.36%
F	18	3.30%	0.33%
G	24	3.30%	0.33%
H	33	2.50%	0.25%
I	40	3.10%	0.31%
J	45	2.50%	0.25%

The most significant finding in the compositional analysis is that the %Pb of solute in the tin-rich phase was above the maximum percentage for equilibrium solidification. A level of 2.5%Pb solute or lower indicates that equilibrium was reached as solidification occurred at the eutectic line of the equilibrium phase diagram. This is the case for the bottom layer of material, which was reheated and cooled from approximately 210°C, and the top of the deposit, which experienced temperatures at approximately 190°C at the deposit surface where the droplet splats solidified relatively slow. It is shown in Table 3 that for areas of deposit that were below the eutectic temperature during deposition, the percentage of lead is significantly higher, illustrating the existence of rapid, non-equilibrium solidification.

5.5 Discussion

The micrographs reveal a finely dispersed rapidly solidified microstructure at locations where the temperature of the deposit at the point of droplet impact was relatively cool ($T < 178°C$). As analysis shows, the solidification occurred completely within the fourth stage discussed in section 5.4. The analysis also shows that when conditions were cool at droplet impact, the percentage of lead solute in the tin rich phase is higher than what is

possible in equilibrium conditions, illustrating the existence of non-equilibrium solidification when the droplet splats are rapidly cooled.

Simulations and SEM analysis reveal that at deposit conditions significantly below the eutectic temperature, rapid solidification occurs within the droplet splat, resulting in fine precipitates and non-equilibrium compositions. The analysis does not, however, duplicate earlier findings which show that a columnar epitaxial microstructure is generated when completely liquid droplets are deposited [16]. This discrepancy can be explained in two ways. Since the droplet thermal model predicts that the droplets are partially solid at the substrate, we would expect the bottom of the deposit to be equiaxed. However, at 40 mm above the substrate surface, the droplets are still completely liquid. At this condition, we would expect columnar epitaxial growth based on the droplet thermal state. Since the deposit at this point is partially liquid, the same mechanisms which occur with a partially liquid droplet could apply here. It is possible that fragmentation of the partially solid dendrites within the deposit would also facilitate equiaxed microstructure. Although the droplet thermal model predicts a liquid droplet at 40 mm above the substrate surface, there is a possibility that the droplets may have in fact been slightly solid. As Appendix G will show, the model is fairly sensitive to certain empirically generated parameters which may vary with different operating conditions. Because the droplets were near the transition point, a small error in cooling rate could have falsely predicted that the droplets were completely liquid instead of slightly solid.

The experiment illustrates the importance of the deposit thermal state. With fairly consistent droplet conditions, the deposit thermal state was able to significantly impact the final grain size of the deposit. The experimental results show the ability to easily modify grain structure through manipulation of the deposit thermal state, illustrating the utility of the UDS process and its potential for creating functionally gradient materials.

Chapter 6

CONCLUDING REMARKS

6.1 Conclusions

Through the combination of modeling and experimentation, several useful insights were gained:

- The deposit thermal model has shown acceptable agreement with experimentation, which justifies the use of the Scheil equation and the enthalpy method. This can be extended to the droplet splat model, provided that a morphology-independent enthalpy model is sufficient. This is the case if solid fragments exist upon droplet impact. Otherwise, solidification will be limited to a single solidification front emerging from the existing solid. In the case of a discrete interface, a solidification front tracking model may be needed to account for the kinetic effects of solidification and the interaction between undercooling and front velocity.

- Conduction through the substrate is the dominant source of heat loss from the deposit. Convection is low in the UDS process since there is no induced gas flow in the chamber. This may or may not be true with other spray forming processes which, as a by-product of the breakup and delivery systems, have high velocity gas impinging on the deposit surface.

- The model has successfully been employed to understand the sensitivity of deposit temperature to mass flux. These studies have shown that it is essential that good control of the spray mass flux distribution be obtained. The exception to this rule is when the deposition rate is low enough that the deposit surface always remains at the substrate control temperature.

- Modeling of the individual droplet splats reveals that the three phases of solidification (droplet solidification; droplet splat solidification; and consolidated deposit solidification), result in significantly different cooling rates. Therefore, by manipulating process parameters to change the percentage of solidification within each of these stages, the resulting microstructure can be adjusted.

- The microstructural evaluation demonstrated that for consistent droplet conditions, the deposit microstructure (grain size) can be varied widely by adjusting the deposit temperature. Since the droplet models show very similar solidification

rates for different deposit temperatures, the difference is in the extent of solidification on the droplet scale. When the deposit is above the eutectic temperature, it is possible that coarsening or liquid phase sintering type phenomena may also be contributing to the variation in microstructure.

6.2 Recommendations for Future Research

Although there has been tremendous strides within the UDS research, several lessons have been learned through this experience.

- For narrow deposits, two dimensional effects significantly change and complicate the process. Transverse thermal gradients cause non-uniform material properties and make modeling and control an order of magnitude more difficult. It is recommended that very wide deposits be generated through multiple orifices to eliminate these effects. Otherwise, two dimensional modeling is required and variable properties in the transverse direction must be tolerated.
- The thermal state of the deposit can be controlled two ways. The first is to reduce the enthalpy flux into the deposit so that the deposit temperature is equal to the control temperature set in the substrate. In this scenario, the conductive cooling from the substrate is far greater than the enthalpy influx, resulting in a deposit with the same temperature as the substrate control temperature. The second method would introduce a significantly higher enthalpy flux to maintain a thermal gradient through the deposit. Although the first approach yields better control, it places limits on the deposit rate. Both approaches provide different thermal conditions which may be necessary for the particular set of microstructure requirements at hand.
- To maintain a desired thermal state during high deposition rates, further steps must be taken to control and monitor the mass flux and spray angle during deposition. The accuracy in both modeling and control has been shown to be a challenge for both the mass flux and the droplet thermal state, which places limits on the control of the deposit thermal state. An ideal system would have an on-line measurement and control system to monitor these parameters and to adjust pressure and charge to maintain the desired spray conditions.

BIBLIOGRAPHY

1. Singer, A.R.E. and Evans, R.W., "Incremental Solidification and Forming," *Metals Technology*, vol. 10, February, pp. 61-68, 1983.

2. Sridharan, K. and Perepezko, J.H., "Microstructure Control in Alloy Steel Powders," International Journal of Powder Metallurgy, vol. 30, no. 3, pp. 301-311, 1994.

3. Leatham, A.G. and Lawly, A., "The Osprey Process: Principles and Applications," *The International Journal of Powder Metallurgy*, vol. 29, no. 4, pp. 321-329, 1993.

4. Lavernia, E.J., Gutierrez, E.M., Szekely, J., and Grant, N.J., "A Mathematical Model of the Liquid Dynamic Compaction Process. Part 1: Heat Flow in Gas Atomization," *International Journal of Rapid Solidification*, vol. 4, pp. 89-124, 1988.

5. Smith, R.W., Novak, R., "Advances and Applications in U.S. Thermal Spray Technology: 1. The Market and R&D," *Powder Metallurgy International*, vol. 23, no. 3, pp. 231-236, 1991.

6. Thorpe, M.L.,"Thermal Spray Industry," Advanced Materials & Processes, pp. 50-61, May 1993.

7. Annavarapu, S. and Doherty, R.D., "Evolution of Microstructure in Spray Casting," *The International Journal of Powder Metallurgy*, vol. 29, no. 4, pp. 331-343, 1993.

8. Liang, X., Earthman, J.C., and Lavernia, E.J., "On the Mechanism of Grain Formation During Spray Atomization and Deposition," *Acta Metallurgica et Materialia*, vol. 40, no. 11, pp. 3003-3016, 1992.

9. Bewlay, B. P. and Cantor, B., "The Relationship between Thermal History and Microstructure in Spray-Deposited Tin-Lead Alloys," *Journal of Materials Research*, vol. 6, no. 7, pp. 1433-1454, 1991.

10. Payne, R.D., Matteson, M.A., and Moran, A.L., "Application of Neural Networks in Spray Forming Technology," *The International Journal of Powder Metallurgy*, vol. 29, no. 4, pp. 345-351, 1993.

11. Acquaviva, P. J., Nowak, T., and Chun, J. H. "Issues in Application of Thermal Spraying to Metal Mold Fabrication," Proceedings of the International Body Engineering Conference (IBEC '94) on Advanced Technologies and Processes, Detroit, Michigan, USA, September 26-29, 1994.

12. Gutierrez-Miravete, E., Lavernia, E.J., Trapaga, G.M., Szekely, J, and Grant, N.J., "A Mathematical Model of the Spray Deposition Process," *Metallurgical Transactions A*, vol. 20A, no. 1, pp. 71-85, 1989.

13. Mathur, P., Apelian, D., and Lawley, A., "Analysis of the Spray Deposition Process," *Acta Metallurgica,* vol. 37, no. 2, pp. 429-443, 1989.

14. Chun, J.-H. and Passow, C.H., "Droplet-Based Manufacturing," *Annals of the CIRP,* vol. 42, no. 1, pp. 235-238, 1993.

15. Passow, C. H., J. H. Chun and T. Ando, "Spray Deposition of a Sn-40 wt pct Pb Alloy with Uniform Droplets," *Metallurgical Transactions A*, 24A:1187-1193, 1993.

16. Chen, C.-A., Chun, J.-H., and Ando, T., "Microstructural Control by Uniform-Droplet Spray Forming," *Proc. 1994 Powder Metallurgy World Congress*, Toronto, Canada, May 1994.

17. Yim, P., "The Role of Surface Oxidation on the Break-up of Laminar Liquid Metal Jets,"PhD. thesis, Department of Mechanical Engineering, M.I.T., 1995.

18. Abel, G.K., "Characterization of Droplet Flight Path and Mass Flux in Droplet-Based Manufacturing," S.M. thesis, Department of Mechanical Engineering, M.I.T., 1995.

19. Passow, C.H., "A Study of Spray Forming Using Uniform Droplet Sprays," S.M. thesis, Department of Mechanical Engineering, M.I.T., 1992.

20. Flemings, M.C., Solidification Processes, McGraw-Hill, New York, pp. 142-161, 1974.

21. Zhang, X. and Athens, A., "A Thermokinetic Model for Rapid Solidification," *International Journal of Rapid Solidification*, vol. 7, pp. 83-107, 1992.

22. Poirier, D.R., and Nandapurkar, P.,"Enthalpies of Binary Alloys during Solidification," *Metallurgical Transactions A*, vol. 19A, pp. 3057-3061, 1988.

23. Fletcher, L.S., "Recent Developments in Contact Conductance Heat Transfer," *Journal of Heat Transfer*, vol. 110, pp. 1059-1070, 1988.

24. Dykhuizen, R.C.,"Review of Impact and Solidification of Molten Thermal Spray Droplets," *Journal of Thermal Spray Technology,* vol. 3(4), pp. 351-361, 1994.

25. Liu, H. and Dandy, D.S., "Deformation and Solidification of Droplets Generated by a Piezo-electric Transducer," Unpublished document, 1994.

26. Jones, H., "Cooling, Freezing, and Substrate Impact of Droplets Formed by Rotary Atomization," *Journal of Physics: Appl. Phys.*, vol. 4, pp. 1657-1660, 1971.

27. Fantassi, S., Vardelle, M., Vardelle, A., and Fauchais, P., "Influence of the Velocity of Plasma Sprayed Particles on the Splat Formation," *Thermal Spray Coatings, Research, Design, and Applications*, ASM International, pp. 1-6, 1993.

28. Clyne, T.W., "Numerical Treatment of Rapid Solidification," *Metallurgical Transactions B*, vol 15., pp. 369-380, 1984.

29. Moreau, C., Lamontagne, M., and Cielo, P., "Influence of the Coating Thickness on the Cooling Rate of Plasma-sprayed Particles Impinging on a Substrate," *Thermal Spray Coatings: Properties, Processes, and Applications*, ASM International, pp. 237-342, 1992.

30. Fukanuma, H., "A Porosity Formation and Flattening Model of an Impinging Molten Particle in Thermal Spray Coatings," *Journal of Thermal Spray Technology,* vol. 3(1), pp. 33-44, 1994.

31. Lavernia, E.J., "The Evolution of Microstructure During Spray Atomization and Deposition," *Journal of Rapid Solidification*, vol. 5., pp 47-85, 1989.

32. Kurz and Fisher, Fundamentals of Solidification, Trans. Tech Publications Ltd, Switzerland, 1984.

33. Gutierrez-Miravete, E., Lavernia, E.J., Trapaga, G.M, and Szekely, J.,"A Mathematical Model of the Liquid Dynamic Compaction Process. Part 2: Formation of the Deposit," *International Journal of Rapid Solidification*, vol. 4, pp. 125-150, 1988.

34. Mathur, P.C.,"Analysis of the Spray Deposition Process," PhD. thesis, Department of Mechanical Engineering, Drexel University, 1988.

35. Edeleson, B.I. and Baldwin, W.M.Jr., "The Effect of Second Phases on the Mechanical Properties of Alloys," *Transactions of the ASM*, vol. 55, pp. 230-250, 1962.

36. Payne, R.D., Moran, A.L., and Cammarata, R.C., "Relating Porosity and Mechanical Properties in Spray Formed Alloy 625," *Ship Materials Engineering Department Research and Development Report*, January, pp. 1-14, 1993.

37. Smith, R.W. and Mutasim, Z.Z , "Plasma Sprayed Refractory Metal Structures and Properties," *Thermal Spray: Research and Applications*, pp. 369-374, 1990.

38. Stover, D., Moulatsiotis, D., and El-Magd, E., "Low Pressure Plasma Sprayed Bulk Superalloy UDIMET 700: Mechanical Properties and Microstructure," *Thermal Spray: Research and Applications*, pp. 375-382, 1990.

39. Benz, M.G., Sawyer, T.F., Clark, F.W., and Dupree, P.L., "Properties of Superalloys Spray Formed Flow Rates of less than 20 cm^3/s," *First International Conference on Spray Forming*, 1990.

40. Kirby, T., Igharo, M., Wood, J.V., and Pratt, R., "Properties of High Speed Steels Made by Spray Forming," *First International Conference on Spray Forming*, 1990.

41. Hribernik, R., Fauland, H.P., Hackl, G., and Kriszt, B., "Properties of the Spray Formed High Speed Steel Grade S 11-2-5-8," *First International Conference on Spray Forming*, 1990.

42. Ebalard, S. and Cohen, M., "Structural and Mechanical Properties of Spray Formed Cast Irons," *First International Conference on Spray Forming*, 1990.

43. Hull, M., "Conference Report: 2nd International Conference on Spray Forming," *International Journal of Powder Metallurgy*, 30(2), pp. 241-246, 1994.

44. Ito, H., Nakamura, R., Shiroyama, M., and Sasaki, T., "Post-Treatment of Plasma Sprayed WC-Co Coatings by Hot Isostatic Pressing," *Thermal Spray: Research and Applications*, pp. 233-238, 1990.

Appendix A

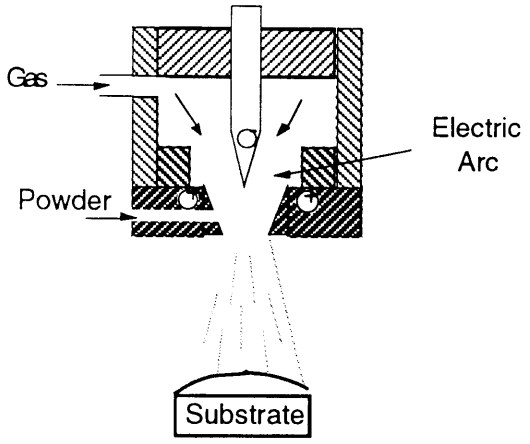
SPRAY FORMING PROCESS COMPARISONS

There are now a variety of thermal spray processes competing to emerge as the most viable method for tool manufacture. As summarized in Table 4 and Fig. 35, these processes differ in their methods of material feeding, heating, and droplet formation.

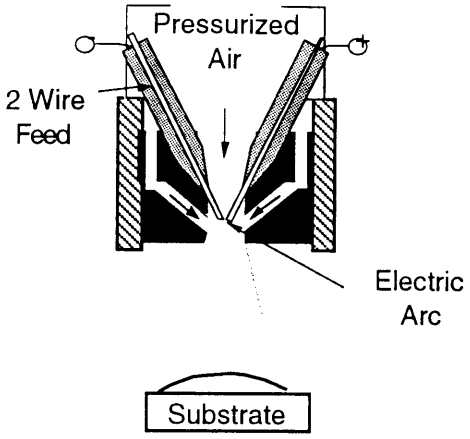
Table 4 - Thermal spray process breakdown

System Feature	Options	Parameters Impacted
Material Feed	• Molten • Wire • Powder	Mass flux and composition
Heating	• Arc • Flame • Furnace/Crucible	Temperature range and thermal control
Droplet Formation	• Gas atomization • Laminar jet breakup	Droplet size and flux distributions and thermal control

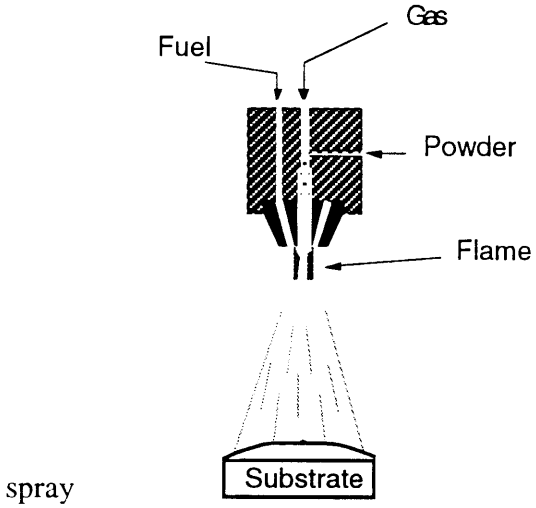
Figs. 35(a) and 35(b) show plasma spray and twin-wire (electric arc) spray, respectively. Both processes use an electric arc. Plasma uses the arc to bring the air or gas to a plasma state. Then powder is injected into the spray and accelerated to the substrate. The arc wire process creates an arc across two wires. The metal then melts off the wire and accelerates to the substrate. Figs. 35(c) and 35(d) show the flame wire and flame powder processes, respectively. The only difference is in the method of feed; the former feeds wire to be melted and the later injects powder to be melted in flight. Figs. 35(e) and 35(f) are similar in that they both use a crucible to melt the metal first. A pressure differential across the orifice causes a stream of molten metal to flow towards the substrate. The difference between the two is in the method of breakup. The gas-atomized spray technique impinges an inert gas into the metal stream, causing atomization to occur. The UDS process imposes a perturbation within the melt to initiate a periodic instability in the laminar, molten jet. The uniform droplets are also electrically charged to repel each other and prevent in-flight agglomeration. Consequently, the uniformity of the droplets is preserved until the droplets reach the substrate.



(a) Plasma spray

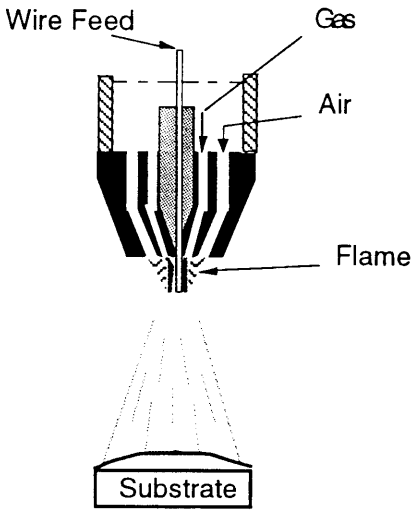


(b) Twin Arc

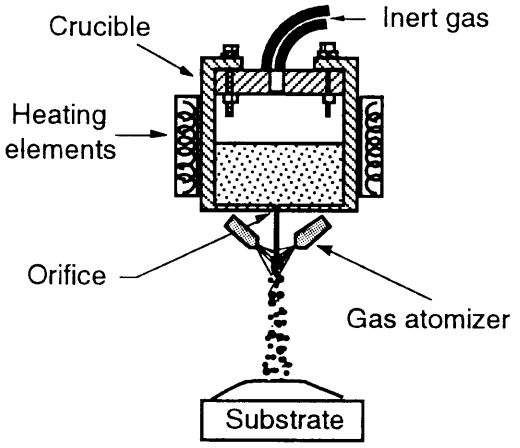


spray

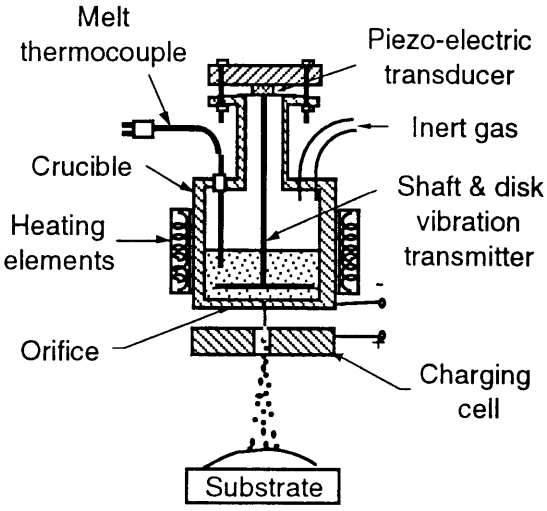
(c) Flame powder spray



(d) Flame wire spray



(e) Gas atomization spray



(f) Uniform droplet spray

Fig. 35. Subgroups of the thermal spray forming process

Appendix B

ISSUES IN THERMAL SPRAY FORMING

The challenge of any thermal spraying system is to control the process parameters to produce the desired deposit properties. The differences among the various spray forming processes, although subtle, result in dramatic differences in process control. The droplet thermal state and mass flux distribution are the most critical parameters for microstructure control. Today's industrial thermal spray processes lack thermal control of the droplets due to the large range of droplet sizes within typical sprays. Non-uniform droplets will heat and cool at different rates resulting in a spray of solid and liquid droplets with large variations in temperature. The non-uniformity of droplet size will also cause variations in droplet trajectories because of the interaction between droplet momentum and aerodynamic drag within the spray chamber. As a result, non-uniform sprays can be controlled in only an approximate sense.

One of the consequences of incomplete process control is the existence of porosity in the sprayed deposit. Porosity is primarily formed by two mechanisms:

1) the entrapment of air between solidified droplets at the deposit, and

2) the existence of inclusions formed when droplets grow thin oxide layers during flight. These oxides do not adhere, forming voids when the solidified deposit is deformed.

Pores within the sprayed material can act as crack initiation sites and can lead to lower ductility [35, 36]. Since the locations and sizes of the pores are typically random, the resulting tool life becomes less predictable. An example of the effect of porosity on ductility is given in Fig. 36. From this figure, it is evident that ductility still suffers at porosities below 1%. Porosity and excessive property variability are both indications of lack of precise process control. Excessive scatter in fatigue properties has also been reported in as-sprayed material [36-43]. If porosity exists, secondary operations such as rolling or hot isostatic pressing (HIP) are necessary [44]. Even with post-processing operations, voids which contain oxidation or contamination can retain crack initiation capability. This will reduce the fatigue strength and predictability of the material. For spray forming to truly become a one step process for geometry, microstructure, and

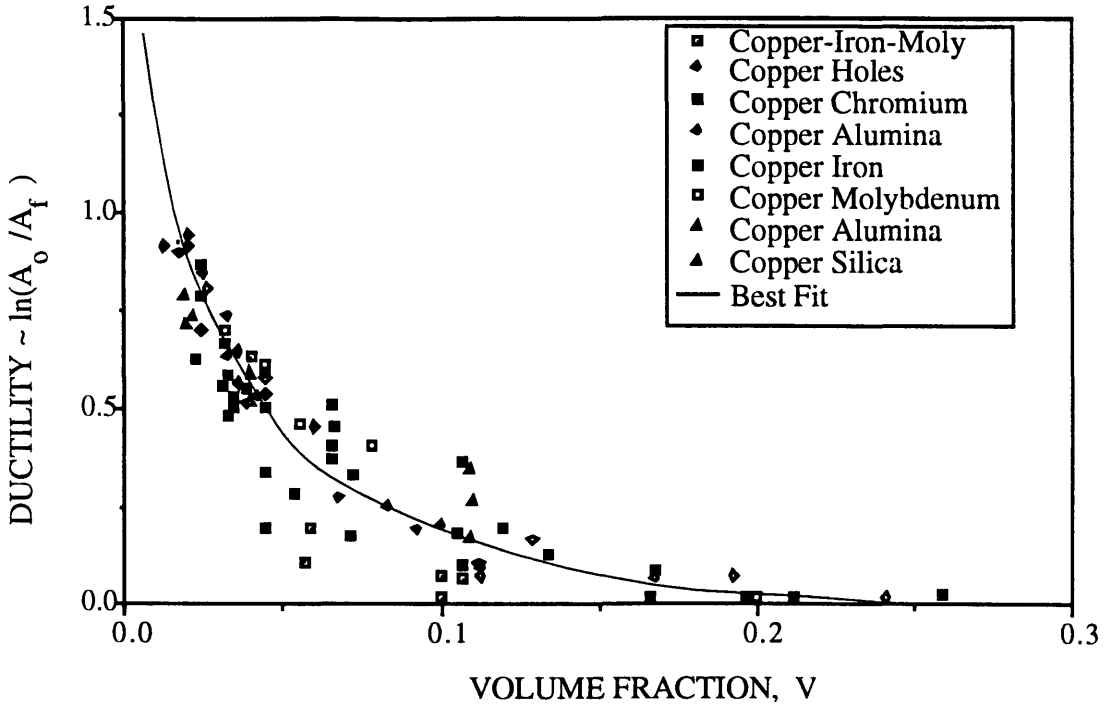


Fig. 36. - Ductility of several copper dispersion alloys vs. volume fraction [24]

surface finish formation, better control must be exhibited to eliminate porosity and increase property reproducibility. For this reason, the UDS process was introduced.

The UDS process produces more uniform droplets and denser deposits than existing spray forming processes. Comparison of reported range in droplet size for various spraying processes are given in Fig. 37. Table 4 compares the reported values of as-sprayed, deposit porosities for various thermal spraying processes including UDS. As shown in Table 5, the range of reported porosity values is significantly large.

Table 5. Summary of reported as-sprayed deposit porosities

Process	**Porosity Range (%)**
Plasma [33-37]	5 - 14
Vacuum Plasma [38-43]	1 - 5
Gas Atomization [44-45]	0.25 - 40
HVOF [49,50]	0.5 - 5
Uniform Droplet Spray	fully dense

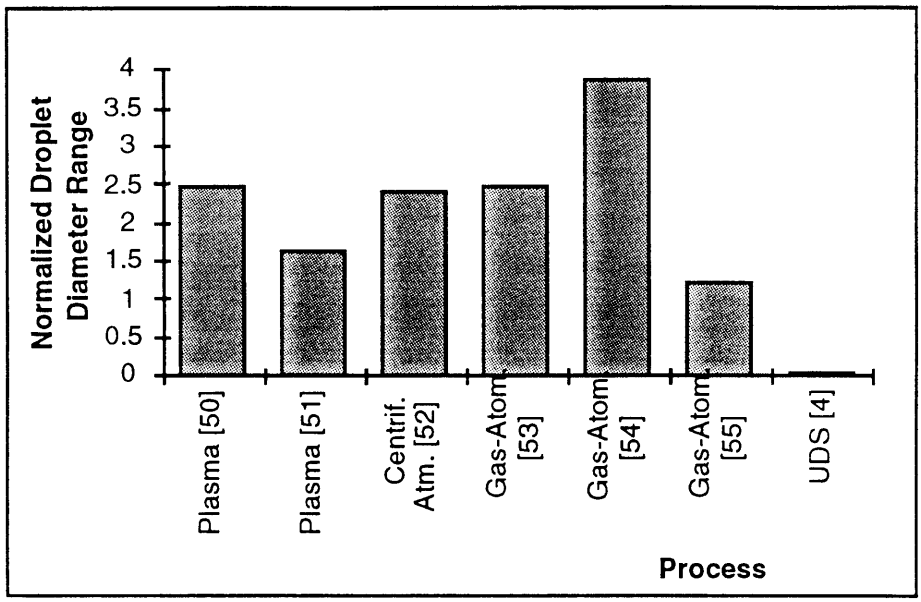


Fig. 37. Comparison of reported range in droplet size for various spraying processes.

Appendix C

BACKGROUND OF DEPOSITION MODELING

Several researchers have modeled the deposit solidification in various thermal spray processes. These models employed explicit, finite difference algorithms with differing solidification models. To model the Liquid Dynamic Compaction process, Gutierrez-Miravete [33] applied a simple straight line enthalpy assumption in the solidification region by assuming:

$$H = \left(\frac{T - T_s}{T_l - T_s}\right) H_f \tag{17}$$

where H is the enthalpy of the deposit, T_s is the solidus temperature, T_l is the liquidus temperature, and H_f is the heat of fusion for the alloy. Mathur [34] employed the Scheil equation assumption, which assumes local equilibrium solidification at the solid/liquid interface to model the Osprey process. Both of these models are considered enthalpy methods. In the modeling of melt spinning, Zhang [21] took the analysis a step further by employing a front tracking algorithm which tracks the location of the solid/liquid interface. This method allows for thermo-kinetic modeling which assumes a relationship between the undercooling of the melt and the interface velocity. The disadvantage of this method is that the model cannot account for dendritic solidification when the mushy region is significant. The enthalpy method, on the other hand, is morphology independent, capable of modeling a large mushy region or a very small interface region.

Appendix D

MODEL DISCRETIZATION

Within the substrate the one dimensional Diffusion Equation is as follows:

$$\rho c_p \frac{\partial T}{\partial t} = \frac{\partial}{\partial x}\left(k(T)\ \frac{\partial T}{\partial x}\right) \tag{18}$$

where K(T) is defined here as conductivity as a function of temperature. At the top substrate surface without any deposit:

$$\rho\ c_p\ \frac{\Delta x}{2}\left(\frac{\left(T_n^{k+1} - T_n^k\right)}{\Delta t}\right)_{surface} = \frac{k\ (T)}{\Delta x}\left(T_{n-1}^k - T_n^k\right) + h\left(T_g^k - T_n^k\right) \tag{19}$$

where Δx is the distance between element centers, T is temperature, k is the iteration number, and n is the element number. Rearranging:

$$T_n^{k+1} = T_n^k + \frac{2\ \Delta t}{\rho\ c_p \Delta x}\left[\frac{k(T)}{\Delta x}\left(T_{n-1}^k - T_n^k\right) + h\left(T_g^k - T_n^k\right)\right] \tag{20}$$

At the top substrate element when deposit exists:

$$\left(\frac{\Delta x}{2}\rho c_p \frac{\left(T_n^{k+1} - T_n^k\right)}{dt}\right) = \left(k(T)\frac{\left(T_{n-1}^k - T_n^k\right)}{dx}\right)_{surface} + h_c\left(T_{n+1}^k - T_n^k\right) \tag{21}$$

Within the substrate, the discretization equation is:

$$\rho c_p \frac{\left(T_n^{k+1} - T_n^k\right)}{\Delta t} = \frac{k(T)}{\Delta x^2}\left(T_{n+1}^k + T_{n-1}^k - 2\ T_n^k\right) \tag{22}$$

Rearranging:

$$T_n^{k+1} = T_n^k + \frac{\Delta t\ k(T)}{\rho c_p \Delta x^2}\left(T_{n+1}^k + T_{n-1}^k - 2\ T_n^k\right) \tag{23}$$

The deposit uses the enthalpy formulation

$$\rho \frac{\partial H}{\partial t} = \frac{\partial}{\partial x}\left(k(T)\ \frac{\partial T}{\partial x}\right) \tag{24}$$

Discretizing the enthalpy equation results in:

$$\rho\frac{\left(H_n^{k+1}-H_n^k\right)}{\Delta t}=\frac{k(T)}{\Delta x^2}\left(T_{n+1}^k+T_{n-1}^k-2\,T_n^k\right) \tag{25}$$

rearranging:

$$H_n^{k+1}=H_n^k+\frac{\Delta t\,k(T)}{\rho\Delta x^2}\left(T_{n+1}^k+T_{n-1}^k-2\,T_n^k\right) \tag{26}$$

At the contact surface with the substrate:

$$H_n^{k+1}=H_n^k+\frac{2\Delta t\,h_c}{\Delta x\rho}\left(T_{n-1}^k-T_n^k\right)+\frac{2\,\Delta t\,k(T)}{\rho\Delta x^2}\left(T_{n+1}^k-T_n^k\right) \tag{27}$$

At the deposit surface during deposition:

$$\rho\frac{\partial(H\Delta x)}{\partial t}=\rho\Delta x\,\frac{\partial H}{\partial t}+\rho H_n\frac{\partial x}{\partial t}=-\left(k(T)\,\frac{\partial T}{\partial x}\right)+h\left(T_g-T_S\right)+\dot{m}H_d \tag{28}$$

To model uneven grid sizes, the following formulation must be employed:

$$\begin{aligned}H_n^{k+1}&=H_n^k+\frac{2\,\Delta t}{\Delta x\,\rho}h\left(T_g^k-T_n^k\right)\\&+\frac{\Delta t\,k(T)}{\rho\dfrac{\Delta x_n}{2}\left(\dfrac{\Delta x_{n-1}+\Delta x_n}{2}\right)}\left(T_{n-1}^k-T_n^k\right)+\frac{2\,\Delta t\dot{m}}{\Delta x\,\rho}\left(H_d-H_n\right)\end{aligned} \tag{29}$$

Appendix E

OPERATING INSTRUCTIONS FOR MODEL

The deposit thermal model input files are listed below:

deposit.c	The ANSI C source file
deposit.out	The executable file - must be compiled on the same system the program will be executed (ie. Sun vs. IBM workstation).
inputs.dat	Contains all input parameter data other than parameters contained within cycles.dat, restart.dat, and enthalpy.dat.
cycles.dat	Contains spray and droplet parameters which can be adjusted with time.
enthalpy.dat	Contains the alloys temperature and corresponding enthalpy and liquid fraction. Allows for any explicit Temperature-Enthalpy relationship.
restart.dat	Contains the initial temperature distribution in the substrate and deposit.

Upon successful modification of the input files, the files must be all placed in the same directory. The program will ask a series of questions and will reprint the inputs from all the input files onto the screen for your review. The following questions are asked:

1) Micro or macro model? - This option will set up the algorithm for the specific case. The macro model assumes the substrate will be modeled and models the contact coefficient between the substrate and the deposit. All heat transfer within the deposit are governed by the deposit conductivity. The micro option models automatically assumes the entire model will be of the deposit material. Any contact resistance between the droplet splat and the preexisting deposit can be modeled with this option. Note that the substrate and deposit thicknesses will be taken from the inputs.dat file. Therefore, the droplet splat for the micro case or the initial deposit thickness for the 'macro' case must be input correctly into the inputs.dat file.
2) Solidification model? - There are three options here.

Scheil Equation - although this option is titled Scheil equation, it really refers to the explicit temperature-enthalpy method, and therefore reads the explicit relationship from the enthalpy.dat file.

Straight Line - This model uses the liquidus temperature in the inputs.dat file and calculates the solidus temperature from the parameter *trange* from the inputs.dat file such that Tsolidus = Tliquidus - Trange. The enthalpy and liquid fraction between these two temperatures are linearly interpolated.

Discrete Interface - This model again uses the explicit T-H relationship in the enthalpy.dat file. However, the algorithm is completely different. The difference is that the solid liquid interface is assumed to be discrete with no mushy region. The model tracks the interface as well as the amount of solute rejected into the liquid. The velocity of the front is calculated and used to determine the undercooling which in turn adjusts the kinetic liquidus temperature. For the time being, a simple V = k*dT equation is used to model the relationship between undercooling and interface velocity.

3) The third question asks whether the run will be a diagnostic run. If this is the case, the output to the screen will be much more extensive in order to track all the critical calculations during the simulation.

Beyond these three questions, the user is required to press return several times to see all the input data printed to the screen. Also, the user is asked to enter pertinent facts or notes about the run. This information will be stored in the enthalpy.out file, which also retains the temperature and liquid fraction history in tabular form for the entire simulation.

The program will return several data files with a variety of data. These are plotted using MATLAB. The plotting files and utility programs exist, these are listed here:

tcplot.m	plots the temperatures at the specified thermocouple locations
plottemp.m	plots the temperatures in constant temperature contours
plotfract.m	plots the liquid fraction in constant liquid fraction contours
rap_enth.c	generates the Temp-Enth-Liquid fraction term assuming the Scheil Equation. Several other features are added to generate relations with variations on this theme.

Appendix F

SIMULATION INPUT FILES

INPUTS.DAT

spray.dat

.92	fld	[fraction]	liquid fraction of the droplet at impact
210	Td	[deg_c]	temperature of droplet at impact
0.08	ddr	[mm/sec]	droplet deposition rate
0.02	diameter	[cm]	droplet deposition rate
36	tdur	[sec]	spray duration - one pass
2.	npass	[-]	number of passes
40.	ppass	[sec]	period of passes
2200.	runtime	[sec]	run duration time

---

sub.dat

748.	ks	[w/m*deg_c]	thermal conductivity of copper
8954.	ds	[kg/m^3]	density of copper
766.2	cps	[J/kg*deg_c]	specific heat for copper
172.	Tc	[deg_c]	substrate control temperature
0.0	flc	[-]	initial substrate liquid fraction
172.	Tdi	[deg_c]	initial deposit temperature
0.0	fldi	[deg_c]	initial deposit liquid fraction
0.030	thsub	[m]	thickness of the copper block
0.004	thdepi	[m]	initial thickness of the deposition

---

tin at ~300 deg K.

60.0	kds1	[w/m*deg_c]	thermal conductivity of tin solid
30.0	kdl1	[w/m*deg_c]	thermal conductivity of tin liquid
7310.	dd1	[kg/m^3]	density of tin
59570.	Hf1	[J/kg]	heat of fusion for tin
243.0	cpd1	[J/kg*deg_c]	specific heat for tin

---

lead props at 300 deg K.

31.4	kds2	[w/m*deg_c]	thermal conductivity of lead
15.4	kdl2	[w/m*deg_c]	thermal conductivity of lead
11340.	dd2	[kg/m^3]	density of lead
26282.	Hf2	[J/kg]	heat of fusion for lead
129.8	cpd2	[J/kg*deg_c]	specific heat for lead

SnPb Alloy ***values adjusted to ge Tm=200

232.00	Tm	[deg_c]	primary melting temperature
328.00	Tm2	[deg_c]	arbitrary melting temperature
183.00	Te	[deg_c]	eutectic Temperature
370	Xe	[-]	eutectic composition of secondary
150	Xt	[-]	fraction of secondary composition (drop)
0.096	kpar	[-]	partition coefficient
0.10	Trange	[deg_c]	range of melting - simple case

---

options.dat

0	restart	[switch]	if 1 - read from restart.dat

0.005	tc1	[W/m^2*deg_c]	plotting location.
0.009	tc2	[W/m^2*deg_c]	plotting location.
0.021	tc3	[W/m^2*deg_c]	plotting location.
0.028	tc4	[W/m^2*deg_c]	plotting location.
0.038	tc5	[W/m^2*deg_c]	plotting location
2.5	hg	[W/m^2*deg_c]	free convection htc.
50.	Tgas	[deg_c]	gas temperature
1000000.0	hc	[W/m^2*deg_c]	contact htc.
10.0	nums	[-]	number of elements in substrate
2.0	numd	[-]	number of elements in init deposit
0.00000001	dt	[sec]	time step
0.0	tinit	[sec]	initial time
.0015	dxi	[m]	deposition element size
20.0	pfreq1	[sec]	frequency of screen printouts
1.0	pfreq2	[sec]	frequency of file printouts
.0009	smsize	[m]	smallest element size
0.003	factor	[m]	dt = factor * stab. criteria
.200	vslope	[m/sec*deg_c]	interface velocity = vslope * undercooling

---

Sn5%Pb at ~400 deg K.

60.89	kds	[w/m*deg_c]	thermal conductivity of tin - solid
60.89	kdl	[w/m*deg_c]	cond liquid - pseudo convective term
7512.	dd1	[kg/m^3]	density of tin
57006.	Hf1	[J/kg]	57006 heat of fusion for tin-lead
1.0	ifl	[J/kg]	scale of Hf
221.78	cpdl	[J/kg*deg_c]	liquid specific heat for tin
221.78	cpds	[J/kg*deg_c]	solid specific heat for tin

243.0	cpds	[J/kg*deg_c]	solid specific heat for tin
235.0	Tliquidus	[deg_c]	
231.0	Tsolidus	[deg_c]	
232.0	Tmelt	[deg_c]	

ENTHALPY.DAT

4 lines **********
239 pairs - temperature vs enthalpy - Scheil Eq. 15%Pb
Scheil Equation Assumed 5-20-95 - added var. from changing Hf
****

239

0.05	-4657.76	0.000
1.09	-4415.05	0.000
2.13	-4172.33	0.000
3.17	-3929.61	0.000
4.21	-3686.89	0.000
.		
.		
.		
183.00	38060.50	0.000
184.04	55797.28	0.377
185.08	56580.37	0.387
186.12	57389.55	0.396
187.16	58226.73	0.407
188.20	59093.97	0.417
189.24	59993.56	0.428
.		
.		
.		
210.03	92309.07	0.896
211.07	95466.54	0.945
212.11	98947.63	1.000
216.26	99917.96	1.000
220.42	100888.83	1.000
224.58	101859.70	1.000
228.74	102830.57	1.000

CYCLES.DAT

cycles.dat - for variable bc's.thi = 0.004"

36 - nctot nc, dur[sec], T[C], lf, ddr[mm/sec]

1	36	200	0.934	0.0273
2	10	200	0.934	0.0
3	35	200	0.936	0.0273
4	10	200	0.936	0.0
5	35	200	0.939	0.0273
6	10	200	0.939	0.0
7	35	200	0.941	0.0273
8	10	200	0.941	0.0
9	35	200	0.943	0.0273
10	62	200	0.943	0.0
11	35	200	0.945	0.0273
12	22	200	0.945	0.0
13	36	200	0.945	0.039
14	10	200	0.945	0.0
15	36	200	0.950	0.043
16	10	200	0.950	0.0
17	32	200	0.952	0.048
18	10	200	0.952	0.0
19	35	200	0.955	0.053
20	9	200	0.955	0.0
21	35	200	0.959	0.055
22	12	200	0.959	0.0
23	35	200	0.965	0.0812
24	15	200	0.965	0.0
25	35	200	0.975	0.09755
26	10	200	0.975	0.0
27	36	200	0.986	0.1139
28	10	200	0.986	0.0
29	36	213	1	0.1235
30	10	213	1	0.0
31	35	213	1	0.1331
32	9	213	1	0.0
33	35	213	1	0.1331
34	10	213	1	0.0
35	35	213	1	0.1331
36	600	213	1	0.0

Appendix G

SPRAY MODEL SENSITIVITIES

In the experiment described in chapters 3 and 6, the droplet spray was adjusted early on in the experiment to provide the correct spray angle. A sensitivity analysis was run at the three charging voltages used during the experiment. Fig. 38 shows that for increased charge, the droplets repel each other more, resulting in an increased spray diameter at the deposit surface.

It was found that the pretest prediction for the spray angle was off by an order of magnitude. Several differences between the experiments used in the initial characterization of the droplet spray [19] and the experiment used in this study include:

1) A different shape leading up to the 100 mm orifice,
2) A new filter slightly upstream of the 100 mm orifice, and
3) A lower pressure drop across the orifice.

These differences resulted in a less turbulent flow which may have resulted in less randomness to the molten jet direction.

The droplet trajectory model assumes there is random disturbances which cause small initial transverse velocities. To match the experimental results observed in the test, the amplitude of the random initial velocity was modified. The sensitivity results given in Fig. 39 indicate that a randomness factor of 0.00010 best matched the observed conditions. This match trajectory model was then used to determine the droplet thermal history. This is necessary, since any error in droplet spray angle will impact the heat transfer conditions on the droplets and therefore change the temperature results. This experience illustrates that any changes in experimental conditions, be it process parameters or apparatus hardware, may yield inaccurate results without further adjustments to the droplet model.

There is clearly a trend that shows that the less turbulent the flow, the smaller the perturbation. Research which could lead to understanding the relationship between initial disturbance and process parameters may prove valuable in obtaining precise control of the droplet spray cone angle or spray diameter at the deposit.

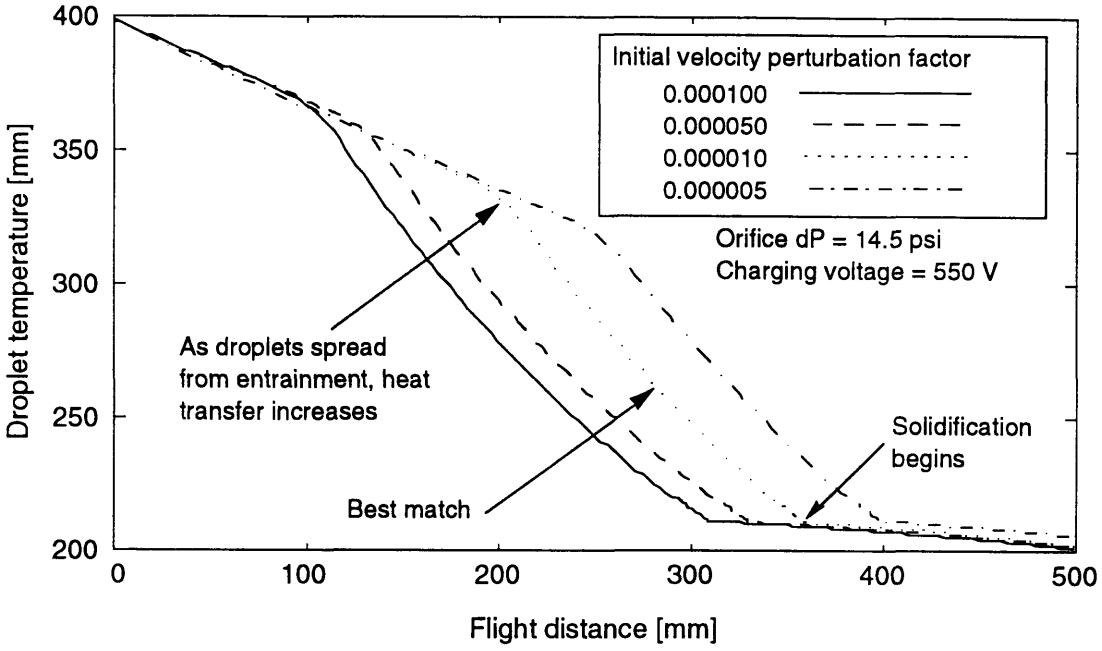


Fig. 38. Droplet temperature sensitivity to instability perturbation

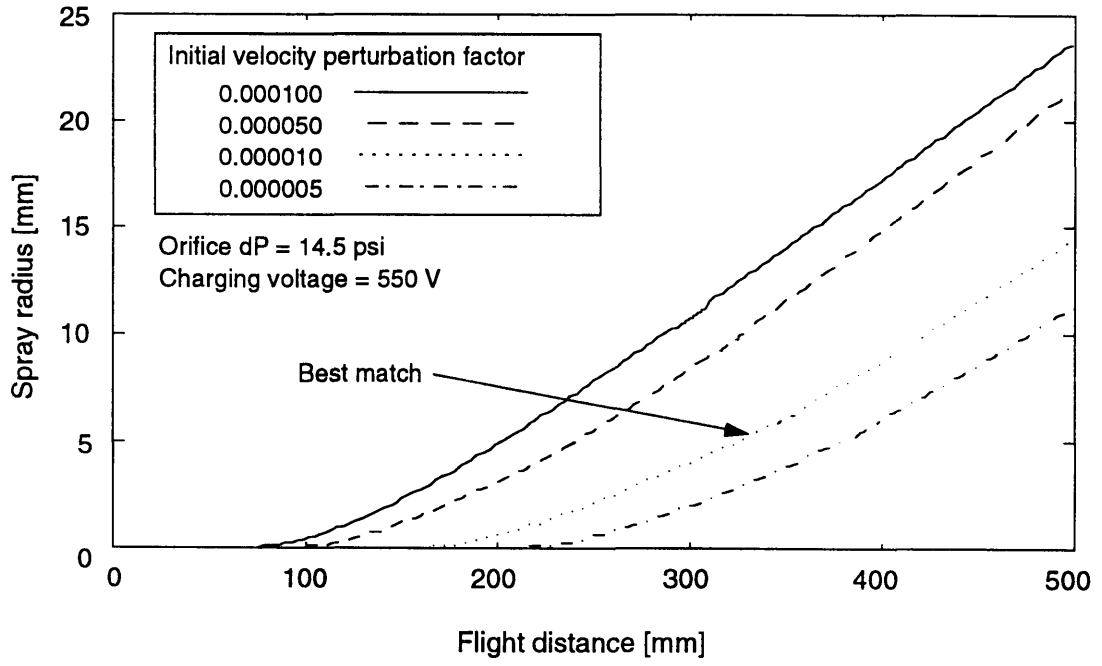


Fig. 39. Spray radius sensitivity to instability perturbation

Appendix H

SIMULATION SOURCE CODE

SIMULATION SOURCE CODE - DEPOSIT.C

```
/*  DBM Substrate and Deposit Thermal Model
    HISTORY - ongoing notes
    3/28/94 - model building commenced
    Initial model will incorporate the governing equations using
    enthalpy in the deposit and temperature in the substrate
    4/02/94   step 1 - start with fixed grid of 30 - 30
    4/26/94 - brought onto athena
    4/27/94 - two tests were successfully performed to verify correctness:
          1 - transient - used high conductivity in model to achieve lumped cap.
          2 - steady - locked the deposition surface to 220. allowed to settle
    4/28/94 - input/output files simple.
    5/05/94 - added functionality with void routines
    6/18/94 - run straight line cp eq. enthalpy model first.
    8/1/94   - giving up on explicit - instability will not leave
    8/10/94 - add implicit routine / source based, linearize solidif.
              velocity over short range include within matrix
    8/13/94 - found error which makes explicit work, FINALLY!
    8/21/94 - adding ando velocity relationship into algorithm
    8/28/94 - successful run of velocity tracking - simple assumption on intvel
   12/28/94 - added T-H relationship for Scheil Equation
    5/3/95   - added side convection to deposit.
*************************************************************************/
#define PI  3.14159
#include <stdio.h>          /* standard library for input/output */
#include <math.h>           /* math library - declares certain functions */
FILE *all;              /* all.dat */
FILE *vv;               /* vel_dt.dat */
FILE *op;               /* enthalpy.dat */
FILE *en;               /* enth.dat */
FILE *td;               /* tempdate */
FILE *ti;               /* title.dat */
FILE *in;               /* inputs8.dat */
FILE *te;               /* temp.dat */
FILE *fr;               /* fraction.dat */
FILE *tm;               /* time.dat */
FILE *sp;               /* space.dat */
FILE *s2;               /* spac2.dat */
FILE *la;               /* lasttemp.dat */
FILE *p1;               /* param1.dat */
FILE *p2;               /* param2.dat */
FILE *uc;               /* undercool.dat */
FILE *vl;               /* intvel.dat */
FILE *pl;               /* pool.dat */
FILE *fxu;              /* flux_up.dat */
FILE *fxd;              /* flux_dn.dat */
FILE *srf;              /* surf.temp */
FILE *tcf;              /* tcsurf.temp */
FILE *tlq;              /* tliq.dat */
FILE *tnt;              /* tint.dat */
FILE *fxi;              /* iflux.dat */
FILE *dtc;              /* dtcont.dat */
FILE *xs;               /* xs.dat */
FILE *xl;               /* xl.dat */
FILE *cy;               /* cycles.dat */
```

```
FILE *tct;              /* tctime.dat */
FILE *tc1;               /* tcplot1.dat */
FILE *tc2;               /* tcplot2.dat */
FILE *tc3;               /* tcplot3.dat */
FILE *tc4;               /* tcplot4.dat */
FILE *tc5;               /* tcplot5.dat */
FILE *sta;              /* restart.dat */
FILE *th;               /* depthick.dat */
FILE *dr;              /* drdr.dat */
FILE *fdr;              /* fldr.dat */
FILE *tdr;              /* tdr.dat */
FILE *hdr;              /* hdr.dat */
/* DECLARE ARRAYS */
double enth[352][3], temp[100], ptemp[100], H[100], PH[100],fl[100],dx[100],x[100],qu[100],qd[100],X[100];
double addr[100], aTd[100], atdur[100],afld[100];
char table[20],filename[20],A[40],B[40],C[40],R[40];
/* DECLARE VARIABLES */
int nctemp,i,iii,vvv,www,j,k,kk,n,mint,m=0,num,numt,nums,numd,ntdrop,opt,numnew,numfin;
int sw,pmode,mode,nsets,swtch,scope,nctot=0,nc=1,restart;
double solid,psolid,c1,c2,time,starttime=0,number,NN,pfreq1,pfreq2,SEterm=0.0,factor,tdelt,VP;
double fluxu,fluxd,mm,dtt,tdur,npass,ppass,p,tinit,Qsub,Qtop,HTOT,PHTOT;
double Hd,Eterm,Qterm,keq1,hg,hc,Tgas,ddr,ddrt,fld,flc,dxi,dxd,dxs,dx1,dx2,dhdx,damp,HTFLUX,QFLUX;
double value,runtime,fltemp,dtsum1,dtsum2,ifl,flsum=1.0;
double dtprint,thin=.000001,diameter;
double tcx1,tcx2,tcx3,tcx4,tcx5,Td,Hf,Tc,dt,pdt=0.0,ppDT,pDT,DT,Tdi,fldi;
double thtot,thsub,thdep,thdepi,dxavg,flux,flt,grnew=0.,dxnew,growth,smsize,ttemp,tswtch,tcycle=0.0;
/* material properties */
double ks,kds,kdl,kpar,ds,dd,cps,cpdl,cpds,Tm2,kds1,kdl1,kds2,kdl2,pv1,pv2,Te,Tm,Trange;
double Tsolidus,Tliquidus,Hs,Hl;
/* Variables related to microscopic model */
double itemp,pitemp,pTiq,Tliq,Xinterr,pXl,dXl,Xs,Xl,Xt,Xe,Xint,vslope,pool,pVEL,VC,ppVEL,VEL;
/* DECLARE FUNCTIONS */
double pinttemp(),inttemp(),HtoT(),TtoH(),FLtoT(),FLsol(),Tsol(),Hsol(),intvel(),kdep(),cpdep();
void gridgrow(),gridcalc(),indata(),manip(),setinit1(),setinit2(),subtemp(),itrack(),fraction();
void depcalc(),depenth(),depenth2(),deptemp(),copytemp(),conc(),allenth();
void depmerge(),dxcalc(),setdt(),isub(),tridiag();
void printa(),printa1(),printa2(),printa3(),printa4(),printa5();
void printb() ,print1(), print2(), print3(), print4(), print5();
void title(),
/****************/
/****************   MAIN Program */
/****************/
main()
{
      /* clear input/output files */
      system("rm *.temp");
           system("rm vel_dt.dat");
           system("rm param1.dat");
           system("rm param2.dat");
           system("cp enth.back enth.dat");
           system("mv hdr.dat hdr.bak");
           system("mv tdr.dat tdr.bak");
           system("mv fldr.dat fldr.bak");
           system("mv drdr.dat drdr.bak");
           /* create all.dat titles */
           all=fopen("all.dat","w");
           fclose(all);
 /* Read in INPUT DATA */
 indata();
 /* initial manipulations of input into correct format - MAKING PHASE DIAGRAM */
 manip();
 /* calculate final grid for plotting purposes and initial run grid (dx[], x[], etc...) */
 gets(table);
 gridcalc();
```

```
/* plots 2 */
print3();
/* Generate Title File */
opt=0;
title();    /*** call titles and comments routine ***/
/* SET Initial Conditions of Temperature, front location, liquid fraction, and Enthalpy */
if(swtch==1 | swtch==3)
 {
   setinit1();
   /********/
  /******** START LOOP */
   /********/
   for(k=1;kk<=1;k++)
        {
         /************ counters ***/
         m=m+1;
         dtsum1=dtsum1+dt;
         dtsum2=dtsum2+dt;
         /************ sets time to be within stability limits **/
         pdt = dt;
         setdt();
         /************ calculate grid size changes       ***/
         dxcalc();
         /********************* Scope = 1 macroscopic ****************/
         if(scope==1)
         {
           depenth();
           for(i=(nums+1);i<=num;i++)
                 temp[i]=HtoT(H[i],fl[i],i);
           subtemp();
           printa();
         /********************* Scope = 2 one drop ****************/
         else if(scope==2)
         {
           allenth();
           for(i=2;i<=num;i++)
                 temp[i]=HtoT(H[i],fl[i],i);
           printa();
         }
         else
          printf("NO SCOPE!!!!!!!!!!!!!!!!!!!!!!!!\n");
          /************ stability ***/
         if(ptemp[num-2]<=(Tc-3))
          if((qu[10]-qd[10])<=100 && (qu[10]-qd[10])>=-100)
           pmode=1;
         /****************** copy new t's and h's to old t's and h's ****************/
         copytemp();
         /************ calculates new thickness ***/
         thtot = thtot + ddr*dt;
         thdep = thdep + ddr*dt;
         /****************** calculate deposition and enthalpy flux *****/
         depcalc();
        }  /*for loop*/
   printa1();
 }     /*if swthfor 1 & 3 */
else if(swtch==2)
 {
   setinit2();
   /********/
  /******** START LOOP */
   /********/
   for(k=1;kk<=1;k++)
        {
         /************ counters ***/
```

```
            m=m+1;
            dtsum1=dtsum1+dt;
            dtsum2=dtsum2+dt;
            /************ sets time to be within stability limits **/
            pdt = dt;
            setdt();
            /************ calculate grid size changes       ***/
            dxcalc();
            /****************** calculate DEPOSIT ENTHALPY ***/
            depenth();
            /****************** calculate concentration *******/
            conc();
            /****************** calculate solidification velocity ***********/
            VEL = intvel();          /* interface velocity  */
            /*********** applying velocity calc from previous iter into this fl change **/
            itrack(VEL);               /* calculates new Xint & vvv based on last dt*/
            /*************** calculate change in liquid fraction for each element **/
            fraction();
            flt = (thtot-Xint)/(thtot-thsub);
            /************ calculate SUBSTRTE TEMPERATURE  ***/
            subtemp();
            /***************** back calculating deposit temperature  ****************/
            for(i=(nums+1);i<=num;i++)
              temp[i]=HtoT(H[i],fl[i],i);
            if(ptemp[num-2]<=(Tc-1))
              if((qu[10]-qd[10])<=100 && (qu[10]-qd[10])>=-100)
                pmode=1;
            /*************** print outs ***************/
            printb();
            /****************** copy new t's and h's to old t's and h's ****************/
            copytemp();
            /************ calculates new thickness ***/
            thtot = thtot + ddr*dt;
            thdep = thdep + ddr*dt;
            /****************** calculate deposition and enthalpy flux *****/
            depcalc();
           }
  }
  else
   printf("NOT A VALID OPTION FOR SOLIDIFICATION MODEL\n");
   /*****************************************  end of while loop */
  /************ print out final run time ****/
  opt=1;
  title();
}
/************************************************** end of main */

/******** Function 1 - Read inputs8.dat ******************/
void indata()
{
  char xxx[50];
  int ii,c,cc,num2;
  double enthalpy,temperature;
  double dd1,Hf1,cpd1,dd2,Hf2,cpd2;
  in = fopen("inputs8.dat","r");
  cy = fopen("cycles.dat","r");
  sta = fopen("restart.dat","r");
  en = fopen("enth.dat","r");
  ti = fopen("title.dat","w");
  fprintf(ti,"INDATA()\n");
  printf("************* MODELLING ASSUMPTION *************\n");
  printf("Type 1 for Macroscopic Model \nType 2 for One Drop Model\n");
  scanf("%1d",&scope);
  printf("scope = %1d\n",scope);
```

```
printf("************ MODELLING ASSUMPTION ************\n");
printf("Type 1 for direct T-H straight line \nType 2 for growth kinetics model\nType 3 for scheil\n");
scanf("%1d",&swtch);
printf("swtch = %1d\n",swtch);
printf("\n************ Print option \n ** enter 0 for normal\n ** enter 1 for diagnostic \n");
scanf("%1d",&pmode);
printf("pmode = %1d\n",pmode);
if(swtch==3)
 {
   /* Read enth.dat */
    printf("reading in temp. vs enthalpy curve \n");
   fscanf(en,"%[^\n]\n",xxx);          /* reading in title #1 */
   fscanf(en,"%[^\n]\n",xxx);          /*  title #2 */
   fscanf(en,"%[^\n]\n",xxx);          /*  title #3 */
   fscanf(en,"%[^\n]\n",xxx);          /*  title #4 */
    fscanf(en,"%d%[^\n]\n",&nsets,xxx);    /* read nsets - number of points on curve */
    printf("nsets = %2d\n",nsets);
   for (ii = 1; ii<=nsets; ii++)
         {
           fscanf(en,"%lf %lf %lf%[^\n]\n",&enth[ii][1],&enth[ii][2],&enth[ii][3],xxx);
           printf(" temperature = %7.2lf, enthalpy = %10.2lf liq. fret = %5.3lf\n",enth[ii][1],enth[ii][2],enth[ii][3]);
         }
    enth[nsets+1][1] = enth[nsets][1];
    enth[nsets+1][2] = enth[nsets][2];
    enth[nsets+1][3] = enth[nsets][3];
   tdelt = enth[2][1] - enth[1][1];
 }
gets(table);
/* Read inputs8.dat */
 /* spray.dat */
fscanf(in,"%[^\n]\n",xxx);
fscanf(in,"%lf%[^\n]\n",&fld.xxx);
fl[0]=fld;
fscanf(in,"%lf%[^\n]\n", &Td.xxx);
fscanf(in,"%lf%[^\n]\n", &ddr.xxx);
fscanf(in,"%lf%[^\n]\n", &diameter.xxx);
fscanf(in,"%lf%[^\n]\n", &tdur.xxx);
fscanf(in,"%lf%[^\n]\n", &npass.xxx);
fscanf(in,"%lf%[^\n]\n", &ppass.xxx);
fscanf(in,"%lf%[^\n]\n", &runtime.xxx);
/* sub.dat */
fscanf(in,"%[^\n]\n",xxx);
fscanf(in,"%[^\n]\n",xxx);
fscanf(in,"%lf%[^\n]\n", &ks.xxx);
fscanf(in,"%lf%[^\n]\n", &ds.xxx);
fscanf(in,"%lf%[^\n]\n", &eps.xxx);
fscanf(in,"%lf%[^\n]\n", &Tc.xxx);
fscanf(in,"%lf%[^\n]\n",&flc.xxx);
fscanf(in,"%lf%[^\n]\n", &Tdi.xxx);
fscanf(in,"%lf%[^\n]\n", &fldi.xxx);
fscanf(in,"%lf%[^\n]\n", &thsub.xxx);
fscanf(in,"%lf%[^\n]\n", &thdepi.xxx);
/* mat.dat tin */
fscanf(in,"%[^\n]\n",xxx);
fscanf(in,"%[^\n]\n",xxx);
fscanf(in,"%lf%[^\n]\n", &kds1.xxx);
fscanf(in,"%lf%[^\n]\n", &kdl1.xxx);
fscanf(in,"%lf%[^\n]\n", &dd1.xxx);
fscanf(in,"%lf%[^\n]\n", &Hf1.xxx);
fscanf(in,"%lf%[^\n]\n", &cpd1.xxx);
/* mat.dat lead */
fscanf(in,"%[^\n]\n",xxx);
fscanf(in,"%[^\n]\n",xxx);
fscanf(in,"%lf%[^\n]\n", &kds2.xxx);
```

```
fscanf(in,"%lf%[^\n]\n", &kdl2.xxx);
fscanf(in,"%lf%[^\n]\n", &dd2.xxx);
fscanf(in,"%lf%[^\n]\n", &Hf2.xxx);
fscanf(in,"%lf%[^\n]\n", &cpd2.xxx);
/* mat.dat alloy */
fscanf(in,"%[^\n]\n",xxx);
fscanf(in,"%[^\n]\n",xxx);
fscanf(in,"%lf%[^\n]\n", &Tm.xxx);
fscanf(in,"%lf%[^\n]\n", &Tm2.xxx);
fscanf(in,"%lf%[^\n]\n", &Te.xxx);
fscanf(in,"%lf%[^\n]\n", &Xe.xxx);
fscanf(in,"%lf%[^\n]\n", &Xt.xxx);
fscanf(in,"%lf%[^\n]\n", &kpar.xxx);
fscanf(in,"%lf%[^\n]\n", &Trange.xxx);
/* options.dat */
fscanf(in,"%[^\n]\n",xxx);
fscanf(in,"%[^\n]\n",xxx);
fscanf(in,"%d%[^\n]\n",&restart.xxx);
fscanf(in,"%lf%[^\n]\n",&tcx1.xxx);
fscanf(in,"%lf%[^\n]\n",&tcx2.xxx);
fscanf(in,"%lf%[^\n]\n",&tcx3.xxx);
fscanf(in,"%lf%[^\n]\n",&tcx4.xxx);
fscanf(in,"%lf%[^\n]\n",&tcx5.xxx);
fscanf(in,"%lf%[^\n]\n",&hg.xxx);
fscanf(in,"%lf%[^\n]\n", &Tgas.xxx);
fscanf(in,"%lf%[^\n]\n", &hc.xxx);
fscanf(in,"%d%[^\n]\n", &nums.xxx);
fscanf(in,"%d%[^\n]\n", &numd.xxx);
fscanf(in,"%lf%[^\n]\n", &dtt.xxx);
fscanf(in,"%lf%[^\n]\n", &tinit.xxx);
fscanf(in,"%lf%[^\n]\n", &dxi.xxx);
fscanf(in,"%lf%[^\n]\n", &pfreq1.xxx);
fscanf(in,"%lf%[^\n]\n", &pfreq2.xxx);
fscanf(in,"%lf%[^\n]\n", &smsize.xxx);
fscanf(in,"%lf%[^\n]\n", &factor.xxx);
fscanf(in,"%lf%[^\n]\n", &vslope.xxx);
/************ Write inputs8.dat to screen **************/
/* spray.dat */
printf("%s\n","-------- spray.dat");
printf("%s = %lf\n","fld", fld);
printf("%s = %lf\n", "Td",Td);
printf("%s = %lf\n", "Hd",Hd);
printf("%s = %lf\n", "ddr",ddr);
printf("%s = %lf\n", "diameter",diameter);
printf("%s = %lf\n", "tdur",tdur);
printf("%s = %lf\n", "npass",npass);
printf("%s = %lf\n", "ppass",ppass);
printf("%s = %lf\n","runtime", runtime);
 /* sub.dat */
printf("%s\n","-------- sub.dat");
printf("%s = %lf\n", "ks",ks);
printf("%s = %lf\n", "ds",ds);
printf("%s = %lf\n", "cps",cps);
printf("%s = %lf\n","Tc", Tc);
printf("%s = %lf\n","flc", flc);
printf("%s = %lf\n","Tdi", Tdi);
printf("%s = %lf\n","fldi", fldi);
printf("%s = %lf\n", "thsub",thsub);
printf("%s = %lf\n", "thdepi",thdepi);
 /* mat1.dat */
printf("%s\n","-------- mat1.dat");
printf("%s = %lf\n", "kds1",kds1);
printf("%s = %lf\n", "kdl1",kdl1);
printf("%s = %lf\n", "dd1",dd1);
```

```
printf("%s = %lf\n", "Hf1",Hf1);
printf("%s = %lf\n", "cpd1",cpd1);
 /* mat2.dat */
printf("%s\n","-------- mat2.dat");
printf("%s = %lf\n", "kds2",kds2);
printf("%s = %lf\n", "kdl2",kdl2);
printf("%s = %lf\n", "dd2",dd2);
printf("%s = %lf\n", "Hf2",Hf2);
printf("%s = %lf\n", "cpd2",cpd2);
 /* alloy.dat */
 printf("%s\n","-------- alloy.dat");
printf("%s = %lf\n", "Tm",Tm);
printf("%s = %lf\n", "Tm2",Tm2);
printf("%s = %lf\n", "Te",Te);
printf("%s = %lf\n", "Xe",Xe);
printf("%s = %lf\n", "Xt",Xt);
printf("%s = %lf\n", "kpar",kpar);
printf("%s = %lf\n", "Trange",Trange);
 /* options.dat */
 printf("%s\n","-------- options.dat");
printf("%s = %d\n", "restart",restart);
 printf("%s = %lf\n", "tex1",tex1);
 printf("%s = %lf\n", "tex2",tex2);
 printf("%s = %lf\n", "tex3",tex3);
 printf("%s = %lf\n", "tex4",tex4);
 printf("%s = %lf\n", "tex5",tex5);
 printf("%s = %lf\n", "hg",hg);
 printf("%s = %lf\n", "Tgas",Tgas);
 printf("%s = %lf\n","hc", hc);
printf("%s = %d\n", "nums",nums);
printf("%s = %d\n", "numd",numd);
 printf("%s = %lf\n","dtt", dtt);
 printf("%s = %lf\n","tinit", tinit);
 printf("%s = %lf\n","dxi", dxi);
 printf("%s = %lf\n","pfreq1", pfreq1);
 printf("%s = %lf\n","pfreq2", pfreq2);
 printf("%s = %lf\n","smsize", smsize);
 printf("%s = %lf\n","factor", factor);
 printf("%s = %lf\n", "vslope",vslope);
 printf("stop and look\n");
 gets(table);
/* manips */
 printf("\nalloy props\n");
pv2 = Xt/dd2 / (Xt/dd2 + (1-Xt)/dd1);
pv1 = 1-pv2;
 kds = kds1*pv1+kds2*pv2;
 kdl = kdl1*pv1+kdl2*pv2;
dd = dd1*pv1+dd2*pv2;
Hf = Hf1*(1-Xt)+Hf2*Xt;
cpds = cpd1*(1-Xt)+cpd2*Xt;
cpdl = cpds;
 printf("%s = %lf\n","pv1", pv1);
 printf("%s = %lf\n","pv2", pv2);
 printf("%s = %lf\n","kds", kds);
 printf("%s = %lf\n","kdl", kdl);
 printf("%s = %lf\n","dd", dd);
 printf("%s = %lf\n","Hf", Hf);
 printf("%s = %lf\n","cpds", cpds);
 fprintf(ti,"%s = %lf\n","kds", kds);
 fprintf(ti,"%s = %lf\n","dd", dd);
 fprintf(ti,"%s = %lf\n","Hf", Hf);
 fprintf(ti,"%s = %lf\n","cpds", cpds);
 gets(table);
 /* Read from restart.dat */
```

```
 if(restart==1)
  {
    printf("reading restart info \n");
    fscanf(sta,"%[^\n]\n",xxx);
    printf("The first line reads %s\n",xxx);
    fscanf(sta,"%d%[^\n]\n",&num2,xxx);
    fscanf(sta,"%lf %[^\n]\n",&starttime,xxx);
   printf("num = %d.  num2 = %d\n",num,num2);
    printf("starttime = %lf\n",starttime);
   if(num2 < (numd+nums) | num2 > (numd+nums))
          {
           printf("ERROR - ELEMENT NO. DOESN'T LINE UP\n");
           fprintf(ti,"ERROR - ELEMENT NO. DOESN'T LINE UP\n");
            gets(table);
          }
   for(e=1; e<=num2; e++)
          {
           printf("ready to scan \n");
            fscanf(sta,"%lf %[^\n]\n",&ptemp[e],xxx);
           if(e>nums)
             TtoH(ptemp[e],e);
           printf("reading restart T=%lf, no = %d, num = %d\n",ptemp[e],e,num);
           fprintf(ti,"reading restart T=%lf, no = %d, num = %d\n",ptemp[e],e,num);
          }
 }

 /* Read from cycles.dat */
 printf("\nreading in cycles info \n");
 fprintf(ti,"\nreading in cycles info \n");
 fscanf(cy,"%[^\n]\n",xxx);
 fscanf(cy,"%d%[^\n]\n",&nctot,xxx);
 printf("nctot = %d\n",nctot);
 fprintf(ti,"nctot = %d\n",nctot);
 for(e=1; e<=nctot; e++)
  {
    fscanf(cy,"%d %lf %lf %lf %lf%[^\n]\n",&ee,&atdur[e],&aTd[e],&afld[e],&addr[e],xxx);
   printf("ne= %3d, atdur= %lf, aTd= %lf, afld= %lf, addr= %lf\n",ee,atdur[ee],aTd[ee],afld[ee],addr[ee]);
    fprintf(ti,"ne= %3d, atdur= %lf, aTd= %lf, afld= %lf, addr= %lf\n",ee,atdur[ee],aTd[ee],afld[ee],addr[ee]);
    addr[e]=addr[e]/1000; /* conv. to m/sec */
  }
 printf("finished reading cycles.dat ok \n");
 gets(table);

 fclose(ti);
 fclose(in);
 fclose(sta);
 fclose(cy);
 fclose(en);
 printf("all indata files closed ok \n");
 gets(table);
}
/****** Calculates the following:
        phase diagram values
         critical enthalpies
          initial compositions
        adj. heat of fusion
******/
void manip()
{
 int e;
  ti=fopen("title.dat","a");
 fprintf(ti,"MANIP()\n");
 printf("in manip()\n");
 mm = (Te - Tm)/Xe;
```

```
Tliquidus = Tm + mm*Xt;
Tsolidus = Te;
if(swtch==1)
 {
  Tliquidus = Tm2;
  Tsolidus  = Tliquidus-Trange;
 }
for(c=1;c<=nctot;c++)
 {
  if(aTd[c]>Tliquidus)
        afld[c]=1.0;
  else if(aTd[c]<Tsolidus)
        afld[c]=0.0;
  else
        aTd[c]=FLtoT(afld[c]);
  Td=aTd[1];
 }
Hs=cpds*(Tsolidus-20);                  /* on for simple method */
Hl=Hs + Hf + cpds*(Tliquidus-Tsolidus);     /* on for simple method */
Hd = TtoH(Td,0);
itemp = Tliquidus;  /* for planar front assumption */
Xs = Xt*kpar;
Xl = Xt;
for(i=www+1;i<=num;i++)
  X[i]=Xt;
printf("\nTliquidus = %7.2lf\nTsolidus  = %7.2lf\n mm = %lf\n Xt =  %lf\n",Tliquidus,Tsolidus,mm,Xt);
printf("Hf = %10.2lf\nHl = %10.2lf\nHs = %10.2lf\nHd= %10.2lf\n\n",Hf,Hl,Hs,Hd);
fprintf(ti,"\nTliquidus = %7.2lf\nTsolidus  = %7.2lf\n mm = %lf\n Xt =  %lf\n",Tliquidus,Tsolidus,mm,Xt);
fprintf(ti,"Hf = %10.2lf\nHl = %10.2lf\nHs = %10.2lf\n\n",Hf,Hl,Hs);
ddr=ddr/1000;       /* convert ddr to m/s from mm/s */
ddrt=ddr;          /* temporary storage */
dt = dtt;
sw=0;
p=0.;
dtsum2=pfreq2;
mode=1;
printf("Hd = %lf\n",Hd);
fprintf(ti,"Hd = %lf\n",Hd);
fclose(ti);
}
/************ Function 2 - Calculation of Final Grid Size */
void gridcalc()
{
  double testtime,newdepth,min,max;
  int realtime;
  char dummy[30];

  ti=fopen("title.dat","a");
  fprintf(ti,"GRIDCALC()\n");
  thtot=thdepi+thsub;          /* total thickness */
  thdep=thdepi;
6.
  /** determining grid size */
  if(thdepi<smsize)
   numd=0;
  else if(thdepi<(smsize+dxi))
   numd=1;
  else
   {
     numd=((thdepi-smsize)/dxi)+1;
   }
  numt = nums + numd;        /* total number */
  num = numt;
  printf("nums= %d, numd= %d, num= %d\n",nums,numd,num);
```

```
  fprintf(ti,"nums= %d, numd= %d, num= %d\n",nums,numd,num);
  ttemp= ptemp[nums];
  /** number of elements at end of run - for plotting */
if(nctot==0)
  newdepth = tdur*ddr*npass;
else
 {
   newdepth=thdepi;
   testtime=0.0;
   for(i=1;i<=nctot;i++)
   {
          newdepth=newdepth+addr[i]*atdur[i];
          testtime=testtime+atdur[i];
          printf("newdepth= %lf, addr[i]= %lf, atdur[i]= %lf testtime= %lf\n",newdepth,addr[i],atdur[i],testtime);
          fprintf(ti,"newdepth= %lf, addr[i]= %lf, atdur[i]= %lf testtime= %lf\n",newdepth,addr[i],atdur[i],testtime);
   }              /* 637 */
 }

  numfin = newdepth/dxi;
  numfin = numfin+1+nums;
  printf("numfin = %d\n",numfin);
  fprintf(ti,"numfin = %d\n",numfin);
  if(runtime>testtime)
    runtime=testtime;
  /* calculate x and dx for initial grid */
  if(scope==1)
   {
      printf("\n***Constant sub.  grid size \n");
      fprintf(ti,"\n***Constant sub.  grid size \n");
     dxs=thsub/((nums)-0.5);
    i=1;
     x[i]= dxs/2.0;
    dx[i]=dxs;
     printf("x[%d]= %lf dx= %lf\n",i,x[i],dx[i]);
     for (i=2;i<=nums;i++)
          {
            x[i]= x[i-1]+dxs;
            dx[i]=dxs;
            printf("x[%d]= %lf dx= %lf\n",i,x[i],dx[i]);
            fprintf(ti,"x[%d]= %lf dx= %lf\n",i,x[i],dx[i]);
          }
   }
  /************ VARYING SIZE ****/
  else if(scope==2)
   {
      printf("\n***Alternative: varying dx\n");
      fprintf(ti,"\n***Alternative: varying dx\n");
    min = 0.0;
    dxs = thsub/nums;
    max = 2*dxs - min;
      dx[1] = 2*(max-(1-0.5)/nums*(max-min))+2*(max-(nums-0.5)/nums*(max-min))-2*dxi;
    x[1]  = 0.0;
    i = 1;
     printf("x[%d]= %lf dx= %lf     c: actual elem thick 1/2 dx \n",i,x[i],dx[i]);
     fprintf(ti,"x[%d]= %lf dx= %lf     c: actual elem thick 1/2 dx \n",i,x[i],dx[i]);
     for (i=2;i<nums;i++)
          {
            dx[i] = max - (i-0.5)/nums*(max-min);
            x[i]= x[i-1]+(dx[i-1]+dx[i])/2;
            fprintf(ti,"x[%d]= %lf dx= %lf\n",i,x[i],dx[i]);
          }
    i= nums;
    dx[i] = 2*dxi;
      x[i]= x[i-1]+(dx[i-1]+dx[i])/2-0.000000001;
```

```
    printf("x[%d]= %lf dx= %lf     c: actual elem thick 1/2 dx \n",i,x[i],dx[i]);
    fprintf(ti,"x[%d]= %lf dx= %lf     c: actual elem thick 1/2 dx \n",i,x[i],dx[i]);
  }
 printf("\n***DEPOSIT GRID dx\n");
 fprintf(ti,"\n***DEPOSIT GRID dx\n");
 if(numd == 0)
  {
    mode = 0;
    x[num+1] = thsub;
    dx[num+1]= dxi;
    i=num+1;
     printf("x[%d]= %lf dx= %lf     c: actual elem thick 1/2 dx \n",i,x[i],dx[i]);
  }
 else
  {
    mode = 1;
    if(numd == 1)
            {
              dx[num]=2*thdepi;
              x[num]=thsub;
              printf("x[%d]= %lf dx= %lf     c: actual elem thick 1/2 dx \n",num,x[num],dx[num]);
              fprintf(ti,"x[%d]= %lf dx= %lf     c: actual elem thick 1/2 dx \n",num,x[num],dx[num]);
            }
    if(numd >= 2)
            {
             dx[nums+1]=dxi;
             x[nums+1]=thsub;
             i=nums+1;
             printf("x[%d]= %lf dx= %lf     c: actual elem thick 1/2 dx \n",i,x[i],dx[i]);
             fprintf(ti,"x[%d]= %lf dx= %lf     c: actual elem thick 1/2 dx \n",i,x[i],dx[i]);
             for (i = (nums+2); i<num ; i++)
              {
                dx[i]=dxi;
                x[i] = x[i-1]+dxi;
                 printf("x[%d]= %lf dx= %lf\n",i,x[i],dx[i]);
                 fprintf(ti,"x[%d]= %lf dx= %lf\n",i,x[i],dx[i]);
              }
             dx[num]= 2*(thtot - (x[num-1]+dx[num-1]/2));
             x[num]= thtot;
             printf("thtot = %lf \n",thtot);
             fprintf(ti,"thtot = %lf \n",thtot);
             i=num;
             fprintf(ti,"x[%d]= %lf dx= %lf     c: actual elem thick 1/2 dx \n",i,x[i],dx[i]);
            }
  }
  x[nums]=thsub-.0000001;
 printf("here are the initial elements\n");
 fprintf(ti,"here are the initial elements\n");
 gets(table);
/* calculate x and dx for final grid */
dx[num+1]=dxi;
x[num+1]=thtot + 0.5*dxi;
for (i=(num+2);i<=(numfin+2);i++)
 {
   dx[i]=dxi;
    x[i]= x[i-1]+dxi;
 }
dxnew = dx[num];                /* top elem. = dxnew; calc. from previous iter */
printf("dxnew = %lf\n\n",dxnew);
fprintf(ti,"dxnew = %lf\n\n",dxnew);
printf("press return to see final grid dimensions\n");
gets(table);
for(i=1;i<=numfin;i++)
  printf("dx[%2i] = %9.7lf x[]= %9.7lf\n",i,dx[i],x[i]);
```

```
  printf("the total number of elements will equal %3i\n\n",(numfin+2));
  fprintf(ti,"the total number of elements will equal %3i\n\n",(numfin+2));
  /*  realtime = runtime/dt/77;
      printf("estimated time to run = %d seconds or %d minutes\n",realtime,(realtime/60)); */
}
/************ Function 3 - Creates titles for output files ****/
void title()
{
  char gg[4],day[4],month[4],nday[4],hr[9],yr[5],name[30];
  char comment1[80],comment2[80],comment3[80],comment4[80];
  ti=fopen("title.dat","a");
 if(opt == 0)
    op=fopen("enthalpy.dat","w");
 if(opt == 1)
    op=fopen("enthalpy.dat","a");
 system("date > tempdate");
 td=fopen("tempdate","r");
  fscanf(td,"%s%s%s%s%s%s%s\n",day,month,nday,hr,gg,yr);
 fclose(td);
 system("rm tempdate");
 if(opt == 1) {
   fprintf(op,"Program Finished: %s %s %s %s %s\n",day,month,nday, hr,yr);
   fprintf(ti,"Program Finished: %s %s %s %s %s\n",day,month,nday, hr,yr);
   fclose(ti);
   fclose(op);
 }
 /* comments for output file  */
 printf("enter name or initials: \n");
 gets(name);
 printf("enter comments (4 lines allowed): \n");
 gets(comment1);
 gets(comment2);
 gets(comment3);
 gets(comment4);
 fprintf(op,"Data generated on: %s %s %s %s %s\n",day,month,nday, hr,yr);
 fprintf(op,"Executed by: %s\n",name);
 fprintf(op,"\nSolidification Option = %d\n",swtch);
  fprintf(op,"\nCOMMENTS:\n%s\n%s\n%s\n%s\n\n",comment1,comment2,comment3,comment4);
 fprintf(ti,"Data generated on: %s %s %s %s %s\n",day,month,nday, hr,yr);
 fprintf(ti,"Executed by: %s\n",name);
 fprintf(ti,"\nSolidification Option = %d\n",swtch);
  fprintf(ti,"\ncomments:\n%s\n%s\n%s\n%s\n\n",comment1,comment2,comment3,comment4);
 /* Assumptions and Headings */
 fprintf(ti,"tgas = %6.2lf, tcontrol = %6.2lf hcontact = %6.3lf hgas = %6.2lf\n\n",Tgas,Tc,hc,hg);
 fclose(ti);
 fclose(op);
 system("cat enthalpy.dat inputs8.dat > enth.dat");
 system("mv enth.dat enthalpy.dat");
  op=fopen("enthalpy.dat","a");
 fprintf(op,"\nnums= %d, dxsub= %10.8lf, dxdepi= %10.8lf, dxsmall= %10.8lf, dxi= %10.8lf\n"
           ,nums,(thsub/nums),(thdepi/numd),smsize,dxi);
 fprintf(op,"\ntime T(i)= nums-2 nums-1   nums nums+1 nums+2 nums+3   n-4   n-3   n-2   n-1     n \n");
 fprintf(op,"-------------------------------------------------------------------------------\n");
 fclose(op);
}
/****** Function 4 - sets initial conditions dep & sub ***/
void setinit1()
{
 int n;
 char dummy[30];
  ti=fopen("title.dat","a");
  printf("\ninitial conditions: \n\n");
 /* SET Initial Condition For Substrate */
 for(n=1;n<=nums; n++)
```

```
 {
   if(restart==0)
          ptemp[n]=Tc;
    temp[n]=ptemp[n];
     printf("ptemp[%2i] = %6.1lf\n",n,ptemp[n]);
   PH[n] = 0.0;
   fl[n] = 0.0;
 }
if(scope==2)
 for(n=1; n<=nums; n++)
   PH[n]=TtoH(Tc,flc);
/* SET Initial Condition For Deposit */
for(n=(nums+1); n<=num; n++)
 {
    printf("set initial deposit, n = %d\n",n);
   if(restart==0)
          {
           ptemp[n]=Tdi;
           fl[n]=fldi;
          }
    PH[n]=TtoH(ptemp[n],n);
     printf("ptemp[%2i] = %6.1lf PH= %7.0lf fl= %4.2lf\n",n,ptemp[n],PH[n],fl[n]);
     fprintf(ti,"ptemp[%2i] = %6.1lf PH= %7.0lf fl= %4.2lf\n",n,ptemp[n],PH[n],fl[n]);
 }
/* set initial droplet state */
n=0;
ptemp[0]=HtoT(Hd,fl[0],n);
temp[0]=ptemp[0];
if(nctot>0)
 {
  ddr   = addr[nc];
  fld   = afld[nc];
  Td    = aTd[nc];
  fl[0]=fld;
  temp[n]=Td;
  ptemp[n]=Td;
  Hd=TtoH(Td,n);
   printf("\n Tliquidus = %lf, Tsolidus = %lf\n",Tliquidus, Tsolidus);
   fprintf(ti,"\n Tliquidus = %lf, Tsolidus = %lf\n",Tliquidus, Tsolidus);
 }
printf("\n droplet temp= %6.1lf, H= %7.0lf, fl= %4.2lf\n",ptemp[0],Hd,fl[0]);
fprintf(ti,"\n droplet temp= %6.1lf, H= %7.0lf, fl= %4.2lf\n",ptemp[0],Hd,fl[0]);
/* stop and check initial conditions */
printf("Press return to continue, control c to stop\n");
gets(table);
printf("\n HERE WE GO ... HOLD ON! \n");
fclose(ti);
}
/****** Function 4 - sets initial conditions dep & sub ***/
void setinit2()
{
 int n;
 char dummy[30];
  printf("\ninitial conditions: \n\n");
 /* SET Initial Condition For Substrate */
 for(n=1;n<=nums; n++)
 {
   ptemp[n]=Tc;
   temp[n]=Tc;
    printf("ptemp[%2i] = %6.1lf\n",n,ptemp[n]);
   PH[n] = 0.0;
   fl[n] = 0.0;
 }
 /* SET Initial Condition For Deposit */
```

```
for(n=(nums+1); n<=num; n++)
 {
   ptemp[n]=Tdi;
   fl[n]=fldi;
   PH[n]=TtoH(ptemp[n],n);
    printf("ptemp[%2i] = %6.1lf PH= %7.0lf fl= %4.2lf\n",n,ptemp[n],PH[n],fl[n]);
 }

/* set initial condition for Xint */
if(num==nums)
 {
   Xint = thsub;
   pool = 0.0;
   vvv=nums+1;
  www=vvv;
    printf("** vvv = %d, Xint= %7.6lf\n\n",vvv,Xint);
 }
 else if(fldi==0.0)
 {
   Xint = thtot;
   pool = 0.0;
   vvv = num-1;
  www=num;
    printf("** vvv = %d, Xint= %7.6lf\n\n",vvv,Xint);
 }
else              /* starts partial liquid as totally liquid */
 {
   Xint = thsub;
    pool = thtot-Xint;
   vvv = nums+1;
  www=vvv;
    printf("** vvv = %d, Xint= %7.6lf\n\n",vvv,Xint);
 }

 /* droplet state */
 ptemp[0]=HtoT(Hd,fl[0],0);
 temp[0]=ptemp[0];
 printf("\n droplet temp= %6.1lf, H= %7.0lf, fl= %4.2lf\n",ptemp[0],Hd,fl[0]);
 /* stop and check initial conditions */
 printf("Press return to continue, control c to stop\n");
 gets(table);
 printf("\n HERE WE GO ... HOLD ON! \n");
}
/************ Function 5 - Set dt for John's stability criteria with factor */
void setdt()
{
 dt = factor*0.5*dx[num]*dx[num]*dd*cpds/(kds+1);
 time = time + dt;
 if (runtime <= time)
   kk=2;
}
/************ Function 6 - calculates expected new dx and grows element when needed ***/
void dxcalc()
{
 dx[num]=dxnew;        /* from previous iteration */
 x[num] = x[num-1]+(dx[num-1]+dxnew)/2;
 if(pmode==1)
   printf("dxcalc()");
 if (mode==0)          /* WAITING for growth to be big enough - only happens once*/
  {
     if(num>(nums+1))
            {
              mode = 1;
              growth = 0.0;
```

```
            }
           growth = thtot - x[num];
           if (growth >= smsize)            /* then grow and go to mode 1 */
            {
              mode = 1;                  /* make a new element */
              gridgrow();
            }
     }
  if (mode==1)          /* top elem growing in size every iteration */
   {
     if ((dx[num]/2)>=(dxi+smsize))    /* if top elem. big enough --> creates new element */
           {
            gridgrow();
           }
     grnew=dt*ddr;
     dxnew=dx[num]+grnew*2;  /* mult by 2 since actual is 1/2 dx */
   }
}
/************ Function 7 - Grid Manipulation - growth by deposition */
void gridgrow()
{
 double dtdx;
 if(swtch == 2)
  {
    if (num==nums)
           {
            printf("ADDING DEPOSITS FIRST ELEMENT with no initial deposit\n");
            temp[nums+1] =temp[nums];
            ptemp[nums+1]=temp[nums];
            fl[nums+1]=flc;
            H[nums+1]=TtoH(temp[nums+1],nums+1);
            PH[nums+1]=H[nums+1];
            dx[nums+1]= 2 * growth;
            x[nums+1]=thsub+dx[nums+1]/2;
            Xint=thsub+(1-fld)*dx[nums+1]/2;
           m=1;
           num=num+1;
           }
    else
           {
            printf("\n\n************** ADDING AN ADDITIONAL ELEMENT*****************");
            printf("ddr= %5.3lf, thsub=%7.5lf, Xint=%10.8lf, thtot=%10.8lf, pooldepth=%10.8lf, dx[num]=%10.9lf\n\n"
                    ,(ddr*1000),thsub,Xint,thtot,(thtot-Xint),dx[num]);
            printf("   Before Element Split:\n   num=%d",num);
            printf("      dx=%10.9lf, x=%10.9lf, ptemp=%7.2lf, H=%7.0lf, fl=%5.3lf\n"
                    ,dx[num],x[num],ptemp[num],PH[num],fl[num]);
            dx[num+1]=dx[num]-2*dxi;
           dx[num]=dxi;
           x[num]=x[num-1]+(dx[num-1]+dx[num])/2;
           x[num+1]=x[num]+(dx[num]+dx[num+1])/2;
           ptemp[num+1]=ptemp[num];
           temp[num+1] =ptemp[num+1];
           ptemp[num] = ptemp[num-1] + (x[num]-x[num-1])/(x[num+1]-x[num-1])*(ptemp[num+1]-ptemp[num-1]);
           temp[num]=ptemp[num];
           /* to account for liquid fraction case */
           fl[num]=(x[num]+dx[num]/2-Xint)/dx[num];
           if(fl[num]>1.0)
             fl[num]=1.0;
           if(fl[num]<0.0)
             fl[num]=0.0;
           fl[num+1]=2*(x[num+1]-Xint)/dx[num+1];
           if(fl[num+1]>1.0)
             fl[num+1]=1.0;
           if(fl[num+1]<0.0)
```

```
            :l[num+1]=0.0;
            PH[num]=TtoH(ptemp[num],num);
            PH[num+1]=TtoH(ptemp[num+1],(num+1));
            printf("   After element split: \n   num=%d ",num);
            printf("     dx=%10.9lf, x=%10.9lf, ptemp=%7.2lf, H=%7.0lf, fl=%5.3lf\n"
                   ,dx[num],x[num],ptemp[num],PH[num],fl[num]);
            printf("   num+1=%d ",(num+1));
     printf("  dx=%10.9lf, x=%10.9lf, ptemp=%7.2lf, H=%7.0lf, fl=%5.3lf\n\n"
              ,dx[num+1],x[num+1],ptemp[num+1],PH[num+1],fl[num+1]);
             printf("*************************************************************************\n\n");
            m=1;
            num=num+1;
           }
  time = time - dt;              /* reset time with new element size */
  kk = 0;
  setdt();
 }
 else /********************** for direct T-H-fl relationship ****************************/
 {
  if (num==nums)
        {
         printf("ADDING DEPOSITS FIRST ELEMENT with no initial deposit\n");
         fl[nums+1]=fld;
         H[nums+1]=Hd;
         PH[nums+1]=H[nums+1];
          temp[nums+1]=HtoT(H[nums+1],fl[nums+1],(nums+1));
          ptemp[nums+1]=temp[nums+1];
         dx[nums+1]= 2 * growth;
          x[nums+1]=thsub+dx[nums+1]/2;
         m=1;
         num=num+1;
        }
  else
        {
         printf("\n\n*************** ADDING AN ADDITIONAL ELEMENT*****************");
          printf("  thsub=%7.5lf, thtot=%10.8lf, pooldepth=%10.8lf, dx[num]=%10.9lf\n\n"
                 ,thsub,thtot,(thtot-Xint),dx[num]);
         printf("   Before Element Split:\n   num=%d",num);
         printf("     dx=%10.9lf, x=%10.9lf, ptemp=%7.2lf, H=%7.0lf, fl=%5.3lf\n"
                ,dx[num],x[num],ptemp[num],PH[num],fl[num]);
         dx[num+1]=dx[num]-2*dxi;
         dx[num]=dxi;
         x[num]=x[num-1]+(dx[num-1]+dx[num])/2;
         x[num+1]=x[num]+(dx[num]+dx[num+1])/2;
         ptemp[num+1]=ptemp[num];
         temp[num+1] =ptemp[num+1];
         fl[num+1]=fl[num];
         PH[num+1]=TtoH(ptemp[num+1],(num+1));
         printf("   After element split: \n   num=%d ",num);
         printf("     dx=%10.9lf, x=%10.9lf, ptemp=%7.2lf, H=%7.0lf, fl=%5.3lf\n"
                ,dx[num],x[num],ptemp[num],PH[num],fl[num]);
         printf("   num+1=%d ",(num+1));
          printf("  dx=%10.9lf, x=%10.9lf, ptemp=%7.2lf, H=%7.0lf, fl=%5.3lf\n\n"
                ,dx[num+1],x[num+1],ptemp[num+1],PH[num+1],fl[num+1]);
           printf("*************************************************************************\n\n");
         m=1;
         num=num+1;
        }
  time = time - dt;              /* reset time with new element size */
  kk = 0;
  setdt();
 }
}
/************ Function 8 - Tracking SolidificationFront */
```

```
void itrack(vel)
    double vel;
{
 double dxint;
 dxint = vel*dt;
 Xint += dxint;
 pool= (thtot-Xint);
 if(Xint>=thtot)
  {
    solid=1.0;
    Xint=thtot;
   VEL=dir;
    vvv= num-1;
   www=num;
   return;
  }
 if(Xint>=(x[num]-dx[num]/2)) /* this statement is now redundant with shifted node pos. */
  {          /* will leave in for now */
    solid=2.0;
    vvv=num-1;
   www=num;
   return;
  }
 if(Xint<=thsub)
  {
    solid=3.0;
    Xint=thsub;
    VEL=0.0;
    vvv = nums+1;
   www=vvv;
   return;
  }
 solid = 5.0;
 if(Xint >= (x[www]+dx[www]/2.0))
  www++;
 else if(Xint < (x[www]-dx[www]/2))
  www--;
 if(Xint >= x[vvv+1])
  vvv++;
 else if(Xint < x[vvv])
  vvv--;
 if(mode==1)
  {
/*  printf("vvv = %d, www= %d, Xint= %lf, VEL= %f, x[vvv]= %lf,x[vvv+1]= %lf iter=%12d\n"
              ,vvv,www,Xint,VEL,x[vvv],x[vvv+1],m);   */
  }
 return;
}
/************ Function 9 - Calculates SUBSTRATE TEMPERATURE  ***/
void subtemp()
{
 double kd,qtermu,qtermd,qterm,cnst,ht,AA,BB,hrad;
 i=1;
 ptemp[i] = Tc;          /* control temp (ideal) */
 temp[i]  = Tc;       /* this could have been put in an initialization
                                 section; leave here for possible control action */
 hrad = 8 + 3*(ptemp[num]-172)/28.0;
 ht = hrad + hg;
 /* substrate internal temperature */
 for(i=2;i<nums;i++)
  {
    cnst = dt/(ds*cps*dx[i]);
     qu[i] = ks/(x[i+1]-x[i])*(ptemp[i+1]-ptemp[i]);
     qd[i] = ks/(x[i]-x[i-1])*(ptemp[i-1]-ptemp[i]);
```

```
    qterm = cnst*(qu[i]+qd[i]);
    temp[i]= ptemp[i] + qterm;
  }
 /* substrate temperature at dep/sub interface */
 cnst = dt/(ds*cps*dx[i]);
 qd[nums] = ks/(x[nums]-x[nums-1])*(ptemp[nums-1]-ptemp[nums]);
 qtermd=cnst*qd[nums];
 if(num>nums)
   {
        qtermu= cnst*hc*(ptemp[nums+1]-ptemp[nums]);
        temp[nums] = ptemp[nums] + 2*qtermu + 2*qtermd;
   }
 else
   {
        qtermu = cnst*ht*(Tgas-ptemp[nums]);
        temp[nums] = ptemp[nums]+2*qtermu+2*qtermd;
   }
 qu[nums] = qtermu/cnst;
 }
/******************* Function 10 - Calculates DEPOSIT ENTHALPY *************/
void depenth()
{
  double Qtermd,Qtermu,Etermalt,Emult,keq2,kd,kd1,kd2,dx1,dx2,qconv,hrad,ht,ddx;
  double afactor[numfin],area[numfin];
  /*convection+radiation */
 if(num>nums)
  {
    qd[nums+1]=  hc * (ptemp[nums]-ptemp[nums+1]);
    Qsub =  dt/(dd*dx[nums+1])*qd[nums+1];
     hrad = 8 + 3*(ptemp[num]-172)/28.0;
    ht = hrad + hg;
    qu[num]= ht*(Tgas-ptemp[num]);
    Qtop  = dt/(dd*dx[num])*qu[num];
    Eterm  =  ddr*dt/dx[num]*(Hd-PH[num]);
    /* Setting up Areas */
     area[nums]=15.5798;
     for(i=nums+1;i<=num;i++)
          {
           ddx  = (x[i]-x[nums]);
           area[i]=15.5798 - 800.2*ddx + 12113.0*ddx*ddx;
           afactor[i] = (area[i]/area[i-1]-1)/2+1;
          }
     /* single deposit element */
    if(num==(nums+1))
          {
           H[num] = PH[num] + 2*Eterm + 2*Qtop + 2*Qsub;  /* all are 2x since size = 1/2 dx */
          }
     /* multiple deposit elements */
    else
          {
           /* at dep/sub interface */
           kd = kdep(ptemp[nums+1],fl[nums+1]);
           kd1 = (kd + kdep(ptemp[i+1],fl[i+1]))/2;
           dx1 =(dx[nums+1]+dx[nums+2])/2;
           qu[nums+1] = kd1/dx1*(ptemp[nums+2]-ptemp[nums+1]);
          Qtermu = dt/(dd*dx[nums+1])*qu[nums+1];
          qconv = dt/(dd*dx[nums+1])*     ht*(Tgas-ptemp[nums+1])*(4*dx[nums+1]/diameter/2);
          H[nums+1] = PH[nums+1] + 2*Qsub + 2*Qtermu + 2*qconv;  /* all are 2x since size = 1/2 dx */
          /* internal to deposit */
          for(i=(nums+2);i<num;i++)
           {
             kd  = kdep(ptemp[i],fl[i]);
             kd1 = (kd + kdep(ptemp[i+1],fl[i+1]))/2;
             kd2 = (kd + kdep(ptemp[i-1],fl[i-1]))/2;
```

```
                dx1 = (dx[i+1]+dx[i])/2;
                dx2 = (dx[i-1]+dx[i])/2;
                qu[i] = kd*(ptemp[i+1]-ptemp[i])/dx1 * afactor[i+1];
                qd[i] = kd*(ptemp[i-1]-ptemp[i])/dx2 / afactor[i];
                qconv = ht*(Tgas-ptemp[i])*(4*dx[i]/diameter);
                H[i] = PH[i] + dt/(dd*dx[i])*(qu[i]+qd[i]+qconv);
              }
             /* deposit surface */
             kd = kdep(ptemp[num],fl[num]);
             dx2 = (dx[num]+dx[num-1])/2;
             qd[num]= kd/dx2*(ptemp[num-1]-ptemp[num]);
             Qtermd = dt/(dd*dx[num])*qd[num];
             H[num] = PH[num] + 2*Eterm + 2*Qtermd + 2*Qtop;  /* the 2 accounts for a 1/2 elem */
            }
  }
 PHTOT = HTOT;  /* HTOT is J/kg * m^3       if HTOT*density = Joules */
 HTOT = 0.5*(H[nums+1]*dx[nums+1] + H[num]*dx[num]);
 for(i=nums+2;i<num;i++)
  HTOT = HTOT+H[i]*dx[i];
 HTFLUX = (HTOT-PHTOT);
 QFLUX = (Eterm+Qtop)*dx[num]/2+Qsub*dx[nums+1];
}
/******************** Function 10 - Calculates DEPOSIT ENTHALPY ************/
void allenth()
{
  double qtermu,qtermd,qterm,cnst,ht,hrad;
  double Qtermd,Qtermu,Etermalt,Emult,keq2,kd,kd1,kd2,dx1,dx2;
 H[1] = TtoH(Tc,0);
 hrad = 8 + 3*(ptemp[num]-172)/28.0;
 ht = hrad + hg;
 /* internal to substrate */
 for(i=2;i<nums;i++)
  {
    kd = kdep(ptemp[i],fl[i]);
    kd1= (kd +kdep(ptemp[i+1],fl[i+1]))/2;
    kd2= (kd +kdep(ptemp[i-1],fl[i-1]))/2;
    dx1 = (dx[i+1]+dx[i])/2;
    dx2 = (dx[i-1]+dx[i])/2;
    qu[i] = kd*(ptemp[i+1]-ptemp[i])/dx1;
              qd[i] = kd*(ptemp[i-1]-ptemp[i])/dx2;
   H[i] = PH[i] + dt/(dd*dx[i])*(qu[i]+qd[i]);
  }
 /* at dep/sub interface - (nums) */
    qu[nums]=  hc * (ptemp[nums+1]-ptemp[nums]);
    dx1 =(dx[nums+1]+dx[nums])/2;
    kd = kdep(ptemp[nums],fl[nums]);
    qd[nums] = kd/dx1*(ptemp[nums-1]-ptemp[nums]);
   Qsub =  dt/(dd*dx[nums])*qd[nums];
   Qtop  = dt/(dd*dx[nums])*qu[nums];
    H[nums] = PH[nums] + 2*Qsub + 2*Qtop;  /* all are 2x since size = 1/2 dx */
 /* DEPOSIT */
   qd[nums+1] = -qu[nums];
   Qsub =  dt/(dd*dx[nums+1])*qd[nums+1];
   qu[num]= ht*(Tgas-ptemp[num]);
   Qtop  = dt/(dd*dx[num])*qu[num];
   Eterm  =  ddr*dt/dx[num]*(Hd-PH[num]);
    /* single deposit element (nums+1 = num) */
   if(num==(nums+1))
          {
           H[num] = PH[num] + 2*Eterm + 2*Qtop + 2*Qsub;  /* all are 2x since size = 1/2 dx */
          }
    /* multiple deposit elements */
   else
          {
```

```
            /* at dep/sub interface (nums+1) */
            kd = kdep(ptemp[nums+1],fl[nums+1]);
            dx1 =(dx[nums+1]+dx[nums+2])/2;
            qu[nums+1] = kd/dx1*(ptemp[nums+2]-ptemp[nums+1]);
            Qtermu = dt/(dd*dx[nums+1])*qu[nums+1];
            H[nums+1] = PH[nums+1] + 2*Qsub + 2*Qtermu;  /* all are 2x since size = 1/2 dx */
            /* internal to deposit */
            for(i=(nums+2);i<num;i++)
             {
               kd = kdep(ptemp[i],fl[i]);
               dx1 = (dx[i+1]+dx[i])/2;
               dx2 = (dx[i-1]+dx[i])/2;
               qu[i] = kd*(ptemp[i+1]-ptemp[i])/dx1;
               qd[i] = kd*(ptemp[i-1]-ptemp[i])/dx2;
              H[i] = PH[i] + dt/(dd*dx[i])*(qu[i]+qd[i]);
             }
            /* deposit surface (num) */
            kd = kdep(ptemp[num],fl[num]);
            dx2 = (dx[num]+dx[num-1])/2;
            qd[num]= kd/dx2*(ptemp[num-1]-ptemp[num]);
            Qtermd = dt/(dd*dx[num])*qd[num];
            H[num] = PH[num] + 2*Eterm + 2*Qtermd + 2*Qtop;  /* the 2 accounts for a 1/2 elem */
           }
}
/***************** Funtion 11 - copy new t's and h's to old t's and h's ****************/
void copytemp()
{
  for(i=2;i<=num;i++)
   {
     ptemp[i]=temp[i];
    PH[i]=H[i];
   }
}
/************ Function 12 - Sets deposition rate  */
void depcalc()
{
  int n=0;

  if(nctot>0)
   {
     tcycle = tcycle + dt;
     if(tcycle>=atdur[nc])
            {
             nc=nc+1;
             tcycle = 0.0;
             ddr   = addr[nc];
             fld   = afld[nc];
             Td    = aTd[nc];
             fl[0]=fld;
             temp[n]=Td;
             ptemp[n]=Td;
             Hd=ToH(Td,n);
              op=fopen("enthalpy.dat","a");
/*           fprintf(op," CYCLE NO.= %d, DEP RATE= %6.1lf, Tdur= %7.0lf\n",nc,ddr,atdur[nc]);
             fprintf(op,"\n Tdrop= %6.1lf, Hd= %7.0lf, fld= %4.2lf\n",Td,Hd,fld); */
            }
   }
  else
   {
     if ( time > (tdur+p*ppass) && sw==0)
            {
             ddr = 0.0;
             p = p + 1.;
             sw=1;
```

```
            if (p==npass)
              sw=2;
            }
    if ( time > (p*ppass) && sw==1)
            {
             ddr=ddrt;
             sw=0;
            }
  }
}
/************* Function 13 - Cp relation for deposit *******/
double cpdep(tmp,fct)
    double tmp,fct;
{
 double cl,cs,c;
 cl = 226.5;                /* liquid   */
 cs = 226.5;                /* solid    */
 c  = fct*cl + (1.0-fct)*cs;   /* mixture  */
 return(c);
}
/************* Function 14 - k relation for deposit *******/
double kdep(tmp,fct)
    double tmp,fct;
{
 double kkkk;
 kkkk  = fct*kdl + (1.0-fct)*kds;  /* mixture  */
 return(kkkk);
}
/**************** Function 15a H to T ********************/
double FLtoT(FL)
    double FL;
{
 double TT;
 /**** simple T to H relationship ********/
 if(swtch == 1)
  {
   if(FL < 0.0 | FL > 1.0)
            {
             printf("ERROR - looking for T outside of fl range\n");
             TT=100000;
            return(TT);
            }
   TT= Tsolidus + FL*(Tliquidus-Tsolidus);
   return(TT);
  }
 else if(swtch == 3)
  {
   TT = FLsol(FL);
/*    if(m>=0 && m<4)
            {
             printf("iter = %3d, nc = %3d, iii = %3d\n",k,nc,iii);
             printf("TT       = %7.2lf,enth[iii]=%7.2lf,enth[iii+1]=%7.2lf\n",TT,enth[iii][1],enth[iii+1][1]);
             printf("fl[nc]=  %7.2lf,enth[iii]=%5.3lf,enth[iii+1]=%5.3lf\n",FL,enth[iii][3],enth[iii+1][3]);
            } */
   return(TT);
  }
}
/**************** Function 15a H to T ********************/
double HtoT(HH,ff,ii)
    double HH,ff;
    int ii;
{
 double TT,Heq;
 /**** simple T to H relationship ********/
```

```
if(swtch == 1)
 {
        if(HHH <= Hs)
         {
           fl[ii] = 0.0;
          TT = 20 + HHH/cpxs;
         }
        if(HHH > Hs&&HHH < Hl)
         {
          fl[ii] = (HHH-Hs)/(Hl-Hs);
          TT= Tsolidus - fl[ii]*(Tliquidus-Tsolidus);
         }
        if(HHH >= Hl)
         {
          fl[ii] = 1.0;
          TT= Tliquidus - (HHH-Hl)/cpdl;
         }
        return(TT);
  }
 /***************** growth kinetics modelling ********************/
 else if(swtch == 2)
  {
   Heq = HH - ff*Hf;
   TT = Tsol(Heq);
   return(TT);
  }
 else if(swtch == 3)
  {
   TT = Tsol(HH);
     fl[ii] = (HH - enth[iii][2])/(enth[iii+1][2]-enth[iii][2]) * (enth[iii+1][3]-enth[iii][3])+enth[iii][3];
   if(m>=0 && m<4)
        {
         printf("iter = %3d, elem no. = %3d, iii = %3d\n",k,ii,iii);
         printf("TT      = %7.2lf,enth[iii]=%7.2lf,enth[iii+1]=%7.2lf\n",TT,enth[iii][1],enth[iii+1][1]);
         printf("HH      = %7.2lf,enth[iii]=%10.2lf,enth[iii+1]=%10.2lf\n",HH,enth[iii][2],enth[iii+1][2]);
         printf("fl[ii]= %7.2lf,enth[iii]=%5.3lf,enth[iii+1]=%5.3lf\n",fl[ii],enth[iii][3],enth[iii+1][3]);
        }
   return(TT);
  }
}
/***************** Function 15b H to T ********************/
double Tsol(HH)
   double HH;
{
 double TT;
  for(k=1;HH<enth[iii][2];k++)
   iii--;
  for(k=1;HH>enth[iii+1][2];k++)
   iii++;
  TT= (HH - enth[iii][2])/(enth[iii+1][2]-enth[iii][2])*tdelt+enth[iii][1];
 return(TT);
}
/***************** Function 15b FL to T ********************/
double FLsol(ff)
   double ff;
{
 double TT;
  for(k=1;ff<enth[iii][3];k++)
   iii--;
  for(k=1;ff>enth[iii+1][3];k++)
   iii++;
  TT= (ff - enth[iii][3])/(enth[iii+1][3]-enth[iii][3])*tdelt+enth[iii][1];
 return(TT);
}
```

```
/***************** Function 16a T to H ********************/
double TtoH(TT,ii)
    double TT;
     int ii;
{
 double HH;
  /**** simple T to H relationship *****/
 if(swtch == 1)
  {
    if(TT <= Tsolidus)
            {
             fl[ii] = 0.0;
             HH = (TT-20)*cpds;
            }
    if(TT > Tsolidus && TT < Tliquidus)
            {
             fl[ii] = (TT-Tsolidus)/(Tliquidus-Tsolidus);
             HH = Hs + fl[ii]*(Hl-Hs);
            }
    if(TT >= Tliquidus)
            {
             fl[ii] = 1.0;
             HH = Hl + (TT-Tliquidus)*cpdl;
            }
    return(HH);
  }
  /***************** growth kinetics modelling ********************/
 if(swtch == 2)
  {
   HH = Hsol(TT) + fl[ii]*Hf;
   return(HH);
  }
 else if(swtch == 3)
  {
   HH = Hsol(TT);
     fl[ii] = (TT - enth[iii][1])/(enth[iii+1][1]-enth[iii][1]) * (enth[iii+1][3]-enth[iii][3])  + enth[iii][3];
   return(HH);
  }
}
/***************** Function 16b T to H ********************/
double Hsol(TT)
    double TT;
{
 double HH;
  int nset;
  nset = (TT-enth[1][1])/tdelt+1;
  HH= (TT - enth[nset][1])/tdelt * (enth[nset+1][2]-enth[nset][2])+enth[nset][2];
 return(HH);
}
/********* DETERMINING CONCENTRATIONS and liquidus temperature *********/
void conc()
{
 /** completely liquid case */
 if(Xl >= Xe)
  {
   Xl = Xe;
   Tliq=Te;
  }
 else if(Xint<=thsub)
  {
   Xl = Xt;
   Xs = kpar*Xl;
   Tliq = Tm + mm*Xl;
  }
```

```
 /**** completely solid case */
 else if(Xint>thtot && VEL >= ddr)
  {
    /* don't change Xs or Xl */
  }
 /** solute redistribution - assumes no solid diffusion, perfect liq. mixing */
 /** sheil eq. equivelent **/
 else
  {
    Xl = (Xl*(thtot-Xint)-Xs*pdt*VEL) / (thtot-(Xint+pdt*VEL));
    Xs = kpar*Xl;
    if(Xl>=Xe)
            {
             Xl=Xe;
             Xs= 1-Xe;
            }
    if( www>nums+1 && www<num )
            X[www]= (fl[www]*X[www]+VEL*pdt/dx[www]*Xs)/(fl[www]+VEL*pdt/dx[www]);
     else if(Xint > thsub && Xint < thtot)
            X[www]= (fl[www]*X[www]+2*VEL*pdt/dx[www]*Xs)/(fl[www]+2*VEL*pdt/dx[www]);
    for(i=www+1;i<=num;i++)
            X[i]=Xl;
    Tliq = Tm + mm*Xl;
  }
}
/************ Function 17 - calculation of velocity front */
double intvel()
{
  double W,vel,aitemp,pitemp,aTliq,pTliq,pXs,VA,VADJ,xx;
/* correction for 1/2 dx elements */
   xx = dx[www];
 if(www==num)
   xx = dx[www]/2;
 if(www==(nums+1))
   xx = dx[www]/2;
 ppDT = pDT;
 pDT = DT;
 ppVEL = pVEL;
 pVEL  = VEL;
  depenth2();
 vel = VEL;
 if(pmode==1)
  {
    printf("Tliq = %lf, Tm = %lf\n mm = %lf, Xl = %lf\n",Tliq,Tm,mm,Xl);
    printf("Tliq = %lf, Tliquidus = %lf\n mm = %lf\n",Tliq,Tliquidus,mm);
    printf("DT = %lf, itemp = %lf, Tliq = %lf \n",DT,itemp,Tliq);
     vv = fopen("vel_dt.dat","a");
    fclose(vv);
  }
 return(vel);
}
/************ Function 18 - Calculation of more precise interface temperature */
double pinttemp()
{
 double TT;
  TT= (Xint - x[vvv])/(x[vvv+1]-x[vvv]) * (ptemp[vvv+1]-ptemp[vvv]) + ptemp[vvv];
 return(TT);
}
/************ Function 18 - Calculation of more precise interface temperature */
double inttemp(xx)
   double xx;
{
 double TT;
  TT= (xx - x[vvv])/(x[vvv+1]-x[vvv]) * (temp[vvv+1]-temp[vvv]) + temp[vvv];
```

```
 return(TT);
}
/********** Function 19 - Fraction - calculate change new liquid fractions */
void fraction()
{
 int xx;
 if(Xint <= thsub)
  {
    for(xx=(nums+1);xx<=num;xx++)
          fl[xx]=1.0;
   return;
  }
 else if(Xint >= thtot)
  {
    for(xx=(nums+1);xx<=num;xx++)
          fl[xx]=0.0;
   return;
  }
 else
  {
    for(xx=(nums+1);xx<www;xx++)
          fl[xx]=0.0;
   fl[www]=((x[www]+dx[www]/2)-Xint)/dx[www];
   if(www==nums+1)
          fl[www]=1-2*(Xint-thsub)/dx[www];
   if(www==num)
          fl[www]=2*(thtot-Xint)/dx[www];
    for(xx=(www+1);xx<=num;xx++)
          fl[xx]=1.0;
   return;
  }
}
/******** Function A - print macro */
void printa()
{
 char dummy[30];
 int xx;
 double tctemp[5][2];
 int tc,nn=0,g;
 if(m>=1&&m<=5)        /* prints to screen */
  {
    printf("m = %d\n",m);
    printa1();
  }
 if(m==3)
  {
    printf("examine growth\n");
   dtsum2=0.0;
  }
 if(dtsum1 >= pfreq1 | pmode == 1)    /* prints to screen */
  {
    printf("\ndeposition rate = %lf mm/s \ndeposit thickness = %lf m \ncycle no = %3.0d \npass no.= %3.0lf\n",(1000*ddr),thdep
    printf("Td = %7.2lf, Hd = %10.2lf, fld = %5.3lf\n",Td,Hd,fld);
    printf("tcycle = %lf sec \ncycles time = %lf sec\n\n",tcycle,atdur[nc]);
    printa1();
    dtsum1=0.0;
  }
 if(dtsum2 >= pfreq2 | pmode == 1)    /* prints to files */
  {
    printa2();
          /********** plotting tc locations ******/
     tcf=fopen("tcsurf.temp","a");
     tct=fopen("tctime.temp","a");
     tc1=fopen("tcplot1.temp","a");
```

```
    tc2=fopen("tcplot2.temp","a");
    tc3=fopen("tcplot3.temp","a");
    tc4=fopen("tcplot4.temp","a");
    tc5=fopen("tcplot5.temp","a");
   th =fopen("depthick.dat","a");
  dr=fopen("drdr.dat","a");
  fdr=fopen("fldr.dat","a");
  tdr=fopen("tdr.dat","a");
  hdr=fopen("hdr.dat","a");
  tctemp[1][1] = tcx1;
  tctemp[2][1] = tcx2;
  tctemp[3][1] = tcx3;
  tctemp[4][1] = tcx4;
  tctemp[5][1] = tcx5;
  printf("time= %lf, nums= %d, num= %d, nn= %d\n",time,nums,num,nn);
  for(tc=1;tc<6;tc++)
      {
        if(tctemp[tc][1]<=thdep)
        {
          nn=0;
          g=0;
          printf(" -- tc= %2d\n",tc);
          while(g==0)
              {
                nn=nn+1;
                if((tctemp[tc][1]+thsub)>=x[nums+nn] && (tctemp[tc][1]+thsub)<=x[nums+nn+1])
                {
                  g=1;
                    tctemp[tc][2]= (tctemp[tc][1]+thsub-x[nums+nn])/(x[nums+nn+1]-x[nums+nn])*(temp[nums+nn+1]-
temp[nums+nn])+temp[nums+nn];
                    printf("nn= %d, tc= %d x= %lf, xtc= %lf",nn, tc, x[nums+nn],(tctemp[tc][1]+thsub));
                    printf(", x2= %lf, t1= %lf, tctemp= %7.2lf, t2=
%lf\n",x[nums+nn+1],temp[nums+nn],tctemp[tc][2],temp[nums+nn+1]);
                }
              }
        }
        else
          tctemp[tc][2]=0.0;
      }
     fprintf(tct,"%lf\n",(time+starttime));
     fprintf(tcf,"%lf\n",temp[num]);
     fprintf(tc1,"%lf\n",tctemp[1][2]);
     fprintf(tc2,"%lf\n",tctemp[2][2]);
     fprintf(tc3,"%lf\n",tctemp[3][2]);
     fprintf(tc4,"%lf\n",tctemp[4][2]);
     fprintf(tc5,"%lf\n",tctemp[5][2]);
     fprintf(th,"%lf\n",(x[num]-thsub));
    fprintf(tdr,"%lf\n",Td);
    fprintf(hdr,"%lf\n",Hd);
    fprintf(dr,"%lf\n",ddr);
    fprintf(fdr,"%lf\n",fld);

   fclose(tcf);
   fclose(tct);
   fclose(tc1);
   fclose(tc2);
   fclose(tc3);
   fclose(tc4);
   fclose(tc5);
  fclose(dr);
  fclose(fdr);
  fclose(tdr);
  fclose(hdr);
   fclose(th);
```

```
    dtsum2=0.0;
   }
  if(temp[num]>=(temp[0]+1))
   {
      printf("\n*_*_*_*_*_*_*\n*_*_*_*_*_*_* Program going unstable\n*_*_*_*_*_*_*\n");
    printa1();
     printf("*_*_*_*_*_*_*\n");
    gets(table);
   }
  if((temp[num]-ttemp) >= 1.)
   {
    printf("temp[num]-ttemp > 1. ttemp = %7.2lf\n",ttemp);
    printa1();
    ttemp = temp[num];
   }
}
/**** printa1 ****/
void printa1()
{
 double dtdx();
 char dummy[30];
 int xx;

 printf(" time = %lf\n iter = %d\n",time,m);
 printf("  i    fl    X   ptemp  temp     dt/dx    PH      H    dx(mm)   x(mm)");
 printf("        qu        qd\n");
  for(xx=num;xx>0;xx--)
     printf("  %3d  %8.6lf %5.3lf %8.4lf %8.4lf %12.2lf %8.1lf %8.1lf %9.5lf %9.5lf %14.3lf %14.3lf\n"
              ,xx,fl[xx],X[xx],ptemp[xx],temp[xx],(temp[xx]-temp[xx-1])/(x[xx]-x[xx-
1]),PH[xx],H[xx],dx[xx]*1000,x[xx]*1000,qu[xx],qd[xx]);
     printf(" thsub=%4.3lf, thtot=%9.8lf, thdep=%9.8lf, dx[num]=%9.8lf, HTFLUX/QFLUX = %7.5lf\n"
            ,thsub,thtot,thdep,dx[num],HTFLUX/QFLUX);
}
/*************** Function B - print out1 - to enthalpy.dat and plot files ***************/
void printa2()
{
  double zero,xtopelem;
   all=fopen("all.dat","a");
   op=fopen("enthalpy.dat","a");
   te=fopen("temp.temp","a");
   fr=fopen("fraction.temp","a");
   s2=fopen("space2.temp","a");
   tm=fopen("time.temp","a");
   pl=fopen("pooldepth.temp","a");
   srf=fopen("surf.temp","a");
  for(i=1;i<=nums;i++)
    fprintf(te,"%8.4lf ",temp[i]);
  for(i=(nums+1);i<=num;i++)
    fprintf(te,"%8.4lf ",temp[i]);
  for(i=(num+1);i<=(numfin+2);i++)
     fprintf(te,"%8.4lf ",1000000000.0);
   fprintf(te,"\n");
  for(i=1;i<=nums;i++)
    fprintf(fr,"%5.3lf ",0.0);
  for(i=(nums+1);i<=num;i++)
    fprintf(fr,"%5.3lf ",fl[i]);
  for(i=(num+1);i<=(numfin+2);i++)
    fprintf(fr,"%5.3lf ",0.0);
  fprintf(fr,"\n");
   fprintf(tm,"%12.10lf\n",(starttime+time));
   fprintf(srf,"%9.4lf\n",ptemp[num]);
  fclose(srf);
  fclose(all);
  fclose(te);
```

```
 fclose(fr);
 fclose(s2);
 fclose(pl);
 fclose(op);
 fclose(tm);
}
/******** Function A - print macro */
void printb()
{
 char dummy[30];
 int xx;
 if(pmode==1)
   gets(table);
 if(solid != psolid)
    printf("psolid= %2lf, solid= %2lf, Xint= %7.6lf, vvv=%d\n",psolid,solid,Xint,vvv);
 psolid=solid;
 if(VEL>1000 | VEL<= (-1000) )
  printf("GETTING BIG --- VEL = %14.0l\n");
 if((Xint-thsub)>=thin)
  {
     printf("(Xint-thsub) >= thin\n");
    print1();
    thin = 100.;
  }
 if(flt==0.0 && tswtch == 0.0)
  {
     printf("flt==0.0 && tswtch == 0.0\n");
    print1();
    runtime = time + 0.0005;
    tswtch = 1.0;
  }
 if(flt<=flsum)
  {
    print1();
      printf("***\n***\n***\n***\n***\n***\n***\n***\n flsum = %lf flt = %lf\n***\n",flsum,flt);
    flsum = flsum - 0.05;
  }
 if(dtsum1 >= pfreq1 | pmode == 1)    /* prints to screen */
  {
    itemp = inttemp(Xint);
   DT = itemp - Tliq;
     printf("\n\n**** Time= %lf(secs), VEL=%10.7lf, DT= %10.7lf, itemp=%14.8lf, Tliq = %9.4lf\n"
            , time, VEL, DT, itemp, Tliq);
    print2();
    dtsum1=0.0;
    if(solid==1.0)
          printf("  completely solid deposit \n");
    else if(solid == 2.0)
          printf("  a thin pool exists \n");
    else if(solid == 3.0)
          printf("  completely liquid deposit \n");
    else if(solid == 4.0)
          printf("  a thin solid layer exists \n");
    else if(solid == 5.0)
          printf("  interface in same element \n");
    else
          printf("  interface has passed thru an element!!\n");
    printf("  i    fl    X   ptemp  temp      dt/dx     PH      H     dx(mm)   x(mm)");
    printf("          qu          qd\n");
     for(xx=num;xx>vvv;xx--)
          printf("  %3d  %8.6lf %5.3lf %8.4lf %8.4lf %12.2lf %8.1lf %8.1lf %9.5lf %9.5lf %14.3lf %14.3lf\n"
                 ,xx,fl[xx],X[xx],ptemp[xx],temp[xx],(temp[xx]-temp[xx-1])/(x[xx]-x[xx-
1]),PH[xx],H[xx],dx[xx]*1000,x[xx]*1000,qu[xx],qd[xx]);
    printf("  VEL= %lf      Tint = %9.5lf    Tliq = %9.5lf    Xint = %12.10lf\n",VEL,itemp,Tliq,Xint);
```

```
    for(xx=vvv;xx>=1;xx--)
        printf("  %3d  %8.6lf %5.3lf %6.2lf %6.2lf %12.2lf %8.1lf %8.1lf %9.5lf %9.5lf %14.3lf %14.3lf\n"
                ,xx,fl[xx],X[xx],ptemp[xx],temp[xx],(temp[xx]-temp[xx-1])/(x[xx]-x[xx-
1]),PH[xx],H[xx],dx[xx]*1000,x[xx]*1000,qu[xx],qd[xx]);
    printf("ddr= %5.3lf, thsub=%4.3lf, Xint=%9.8lf, thtot=%9.8lf, pooldepth=%9.8lf, dx[num]=%9.8lf, HTFLUX/QFLUX = %7.5l
            ,(1000*ddr),thsub,Xint,thtot,pool,dx[num],HTFLUX/QFLUX);
        printf("*****************************************************************************\n");
  }
 if(m>=1&&m<=5)        /* prints to screen */
  {
    print2();
  }
 if(m==3)
  {
    printf("examine growth\n");
   /*     gets(table); */
  }
 if(temp[num]>=temp[0])
  {
      printf("\n*_*_*_*_*_*_*\n*_*_*_*_*_*_* Program going unstable\n*_*_*_*_*_*_*\n");
    print2();
     printf("*_*_*_*_*_*_*\n");
    gets(table);
  }
 if((temp[num]-ttemp) >= 1.)
  {
    printf("* num= %d, t=%7.6lf T= %7.2lf itemp= %7.2lf Xint=%12.9lf S=%2.0lf,\nthtot= %8.7lf vvv= %d fl= %4.2lf H= %7.0lf
            ,num,time,temp[num],itemp,Xint,solid,thtot,vvv,fl[num],H[num]);
    /* print5(); */
    ttemp = temp[num];
  }
 if(dtsum2 >= pfreq2 | pmode==1)    /* prints to files  */
  {
    print1();
    dtsum2=0.0;
  }
}
/*************** Function B - print out1 - to enthalpy.dat and plot files ***************/
void print1()
{
 double zero,xtopelem;

  all=fopen("all.dat","a");
  op=fopen("enthalpy.dat","a");
  te=fopen("temp.temp","a");
  fr=fopen("fraction.temp","a");
  s2=fopen("space2.temp","a");
  tm=fopen("time.temp","a");
  vl=fopen("intvel.temp","a");
 uc=fopen("undercool.temp","a");
  pl=fopen("pooldepth.temp","a");
 fxu=fopen("fluxu.temp","a");
 fxd=fopen("fluxd.temp","a");
 srf=fopen("surf.temp","a");
 tlq=fopen("tliq.temp","a");
 tnt=fopen("tint.temp","a");
 fxi=fopen("iflux.temp","a");
 dtc=fopen("dtcont.temp","a");
 xs=fopen("xs.temp","a");
 xl=fopen("xl.temp","a");
 vv = fopen("vel_dt.dat","a");
 for(i=1;i<=nums;i++)
   fprintf(te,"%8.4lf ",temp[i]);
 for(i=(nums+1);i<=num;i++)
```

```
    fprintf(te,"%8.4lf ",temp[i]);
  for(i=(num+1);i<=(numfin+2);i++)
     fprintf(te,"%8.4lf ",1000000000.0);
  fprintf(te,"\n");
  fprintf(tm,"%12.10lf\n",(starttime+time));
  fprintf(uc,"%12.10lf\n",DT);
  fprintf(vl,"%12.10lf\n",VEL);
  fprintf(pl,"%12.10lf\n",pool);
  fluxu = kdep(ptemp[num],fl[num])*(ptemp[num]-ptemp[num-1])/(x[num]-x[num-1]);
  fluxd = ks*(ptemp[nums]-ptemp[nums-1])/dx[nums-1];
 flux = dd*(PH[www]-H[www])*dx[www]/(dt*Hf);
  fprintf(fxu,"%12.10lf\n",fluxu);
  fprintf(fxd,"%12.10lf\n",fluxd);
  fprintf(srf,"%9.4lf\n",ptemp[num]);
  fprintf(tlq,"%9.4lf\n",Tliq);
  fprintf(tnt,"%9.4lf\n",itemp);
  fprintf(dtc,"%9.4lf\n",(ptemp[nums+1]-ptemp[nums]));
  fprintf(fxi,"%9.4lf\n",flux);
  fprintf(xs,"%9.7lf\n",Xs);
  fprintf(xl,"%9.7lf\n",Xl);
 fclose(srf);
 fclose(tlq);
 fclose(tnt);
 fclose(dtc);
 fclose(xs);
 fclose(xl);
 fclose(op);
 fclose(all);
 fclose(te);
 fclose(fr);
 fclose(s2);
 fclose(tm);
 fclose(uc);
 fclose(vl);
 fclose(pl);
 fclose(fxi);
 fclose(fxu);
 fclose(fxd);
 fclose(vv);
}
/*************** Function C - print out2 - to screen  ***************/
void print2()
{
  printf("\n**** mode= %d, smsize= %8.6lf, growth= %14.12lf, ",mode,(1000*smsize),(1000*growth));
  printf("grnew= %14.12lf, dxnew= %14.12lf, ddr= %12.10lf\n",(1000*grnew),(1000*dxnew),(1000*ddr));
 printf("  it    = %10d,   new it =    %7d,   dt = %14.12lf\n\n",k,m,dt);
 printf("  ppVEL = %10.6lf,    pVEL = %10.6lf,   VEL = %10.6lf,       VC=%12.10lf\n",ppVEL,pVEL,VEL,VC);
 printf("  Tliq  = %10.5lf,   itemp  = %10.5lf,   vvv = %d, www= %d\n",Tliq,itemp,vvv,www);
 printf("  ppDT  = %10.6lf,      pDT = %10.6lf,    DT = %10.6lf\n",ppDT,pDT,DT);
 printf("  Xs    = %10.8lf,       Xl = %10.8lf,  Xint = %10.8lf,   Xinterr=%lf\n",Xs,Xl,Xint,Xinterr);
  printf("  kpar  = %10.8lf,  dt/pdt  = %10.7lf\n\n",kpar,(dt/pdt));
}
/*************** Function D - print out3 - to plot files ***************/
void print3()
{
  sp=fopen("space.temp","w");
 printf("EXECUTING PRINT3\n");
 for(i=1;i<=nums;i++)
    fprintf(sp,"%12.7lf\n",x[i]);
 for(i=(nums+1);i<num;i++)
    fprintf(sp,"%12.7lf\n",x[i]);
  fprintf(sp,"%12.7lf\n",x[num]);
 for(i=(num+1);i<=(numfin+2);i++)
    fprintf(sp,"%12.7lf\n",x[i]);
```

```
  fclose(sp);
}
/*************** Function E - print out4 - to plot files ***************/
void print4()
{
   la=fopen("lasttemp.dat","a");
  printf("EXECUTING PRINT4\n");
  for(i=1;i<=num;i++)
     fprintf(la,"%12.4lf\n",temp[i]);
  fclose(la);
}
/*************** Function F - print out1 - to enthalpy.dat and plot files ***************/
void print5()
{
   p1=fopen("param1.dat","a");
   p2=fopen("param2.dat","a");
    fprintf(p1,"%15.13lf\n",time);
    fprintf(p2,"%7.2lf\n",temp[nums]);
  fclose(p1);
  fclose(p2);
}
```